Summer 8-8-2024

Mitigating Class Imbalance in Time Series Classification via Generative Modeling

Yang Chen

Recommended Citation
Chen, Yang, "Mitigating Class Imbalance in Time Series Classification via Generative Modeling." Dissertation, Georgia State University, 2024.
doi: https://doi.org/10.57709/37354731

Mitigating Class Imbalance in Time Series Classification via Generative Modeling

by

Yang Chen

Under the Direction of Rafal Angryk, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2024

ABSTRACT

Most classification techniques assume a uniform distribution of training data classes. However, well-balanced data is rare, with infrequent events often being the most valuable. This well known class imbalance issue severely hinders the performance of supervised algorithms, limiting their ability to accurately predict minority classes and resulting in analyses that lack practical operational value. Generative models in machine learning are designed to learn the underlying patterns or distribution of existing data, enabling them to generate new data that resembles the original dataset. Inspired by the success of Generative Adversarial Networks (GANs) in synthetic image generation, we employed Conditional GAN (CGAN) to generate synthetic multivariate time series data on a benchmark dataset for solar flare forecasting. Our experiments show that the synthetic data produced is statistically comparable to real data and and effective in addressing class imbalance. However, CGANs, typically trained on well-processed and balanced datasets like MNIST and CIFAR-10, face challenges when trained with imbalanced datasets. These challenges can negatively impact CGAN performance, reducing the quality of synthetic samples, especially for minority class(es). To handle this issue, we propose a Two-stage CGAN framework that enhances the quality and diversity of synthetic samples for minority classes in both image and time series generation tasks. We also introduce FFAD, a novel internal evaluation metric specifically designed to assess the fidelity of synthetic time series data and evaluate generative model performance. Our comprehensive evaluation on solar flare forecasting demonstrates the efficacy of our CGAN-based approach in mitigating class imbalance issues, highlighting its potential to enhance predictive capabilities for rare but critical events across various domains.

INDEX WORDS: Class imbalance, Synthetic data generation, Generative adversarial networks (GANs), Multivariate time series, Evaluation metric, Solar flare forecasting

Mitigating Class Imbalance in Time Series Classification via Generative Modeling

by

Yang Chen

Committee Chair: Rafal Angryk

Committee: Dustin Kempton

Berkay Aydin

Viacheslav Sadykov

Electronic Version Approved:

Office of Graduate Services

College of Arts and Sciences

Georgia State University

August 2024

DEDICATION

To my beloved parents, whose unwavering support, encouragement, and love have been the foundation of my journey. Your endless belief in my potential has given me the strength and determination to pursue my dreams. I am forever grateful for your wisdom, kindness, and for being my greatest and eternal source of inspiration.

ACKNOWLEDGMENTS

The work presented in this dissertation was made possible with the support of many people. First and foremost, I extend my heartfelt gratitude to my advisor, Dr. Rafal Angryk, for providing me with the opportunity to study in the U.S. and for guiding me throughout my Ph.D. studies. Your mentorship has been crucial in helping me navigate key milestones, and I am deeply thankful for all you have done.

Second, I would like to express my appreciation to Dr. Dustin Kempton. Your intelligence and invaluable insights have been instrumental in shaping my research. Your guidance, expertise, and support have profoundly enriched my academic journey. Thank you for your dedication and patience. This dissertation would not have been possible without your exceptional mentorship.

I would also like to thank Dr. Berkay Aydin and Dr. Viacheslav Sadykov for serving on my Advisory Committee and supporting my doctoral training. Your expertise and constructive feedback have greatly contributed to the quality and depth of this dissertation.

Furthermore, I appreciate all the discussions, feedback, and friendship from the members of our lab over the years — Azim, Ruizhe, Xumin, Junzhi, Annie, Chetraj, and others. Working at DMLab in the field of solar physics has provided me with the invaluable opportunity to tackle real-world problems similar to those in the industry, significantly aiding in landing my job and advancing my career. It has been a great honor to work here.

Funding Acknowledgement

This project has been supported in part by funding from the Division of Advanced Cyberinfrastructure within the Directorate for Computer and Information Science and Engineering, the Division of Atmospheric & Geospace Sciences within the Directorate for Geosciences, under NSF awards #1931555 and # 1936361. It has also been supported by NASA's Space Weather Science Application Research-to-Operations-to-Research program grant #80NSSC22K0272.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Living With a Star (LWS) is a NASA initiative that studies the Sun-Earth system and its effects on human and societal life. NASA launched the Solar Dynamics Observatory (SDO) mission in February 2010 as part of its commitment to this programme. The SDO mission is a vital device for studying solar activity, which might result in hazardous space weather. This space weather activity has the potential to have devastating effects on space and air travel, power grids, GPS, and communications satellites [3]. In March of 1989, for instance, geomagnetically generated currents, formed when charged particles from a coronal mass ejection impacted the earth's atmosphere, led to blackouts and direct expenses of tens of millions of dollars for the Canadian electric utility Hydro-Quebec [4]. If a similar occurrence had occurred during the summer months, it would have likely caused widespread blackouts in the northeastern United States, resulting in billions of dollars in economic damage [4].

A solar flare is an event that occurs in the solar corona and is characterised by a sudden orders-of-magnitude brightening in Extreme Ultra-Violet (EUV) and X-ray emissions, and for large flares, gamma-ray emissions, from a tiny location on the Sun lasting from minutes to hours. Since 1974, Geostationary Operational Environmental satellites (GOES) operated by the National Oceanic and Atmospheric Administration (NOAA) have been used to automatically detect and categorise X-ray flares into wavelength bands of 1-8 Ångstrom. According to the peak soft X-ray flux in this range, flares are logarithmically characterised as A, B, C, M, and X, from weaker to stronger. Fig. 1.1 displays an example GOES satellite X-ray flux

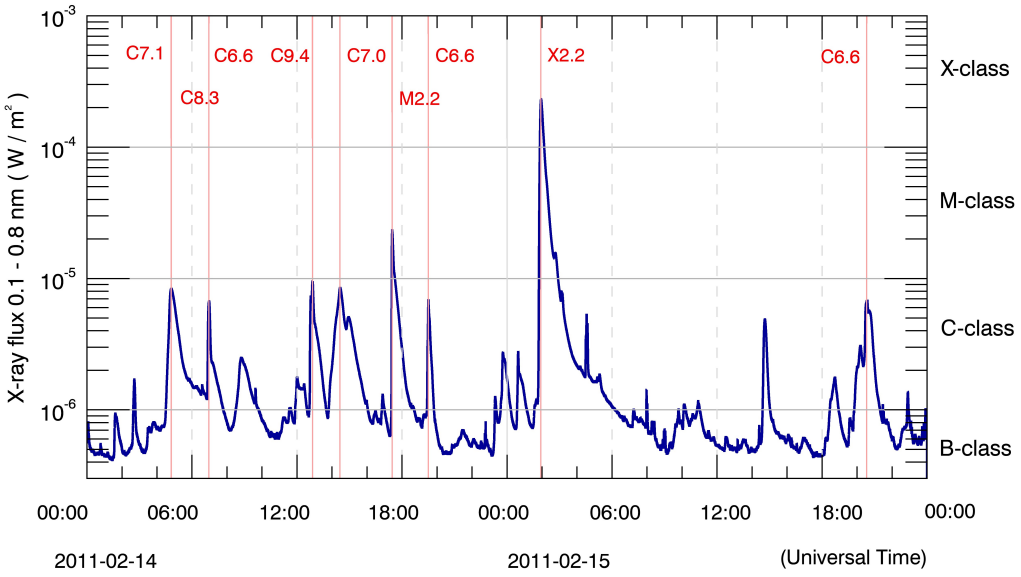

Figure 1.1 GOES15 1 - 8 Å solar X-ray flux from 2011-02-14 to 2011-02-15. The GOES flare classification is provided on the minor y-axis. The plot also includes annotations of flares exceeding GOES class C5.0, with red vertical lines indicating the flares' peak time. The example interval also shows that during these two days of intense activity, background X-ray flux was high, making it difficult to identify small flares.

series annotated with certain flare events based on the peak X-ray flux. In a typical binary classification strategy, the most intense flares, namely the M and X classes, are identified as the positive class. In contrast, no flare occurrence and flares of A, B and C classes are identified as the negative class.

Natural sciences, particularly solar physics, face unique challenges stemming from the rarity of the events of interest, high dimensionality of observational data, and behavior of the systems being observed. Despite meticulous curation, data collected for real-world prob-

lems rarely comes in a clean, ready-to-use format. Inherent challenges, whether arising from the subject matter itself or the data collection methodology, must be identified, understood, and addressed appropriately. Solar flare forecasting is an excellent example of analyzing rare-events (i.e., flaring events) of interest. In this dissertation, we focus on the class imbalance caused by the rarity of flaring events and its effects on data preparation for machine learning (ML) methods. By addressing these challenges, we aim to enhance the accuracy and reliability of ML models in solar physics and other natural science domains.

1.1 Motivation

Class imbalance refers to a situation where the instances of one or more classes in a dataset are significantly fewer than those of other classes. While data experts are aware of these challenges, the explicit impact on analysis and subsequent decision-making is often overlooked. Addressing class imbalance between majority and minority classes is crucial, as significant imbalance can introduce bias towards majority classes, leading to unsatisfactory results for any classifier [5; 6; 7]. In solar flare prediction, the lack of positive flare data is a notable issue that must be adequately addressed. This study aims to mitigate the class-imbalance issue by generating synthetic multivariate time series data using generative adversarial networks (GANs) based on deep learning techniques.

The well-known class-imbalance issue can also profoundly affect training GANs. The authors of [8] state that traditional GANs cannot generate minority-class images from imbalanced datasets. Few studies address this imbalance at the algorithm level, either by

employing autoencoders to learn latent features or by modifying objective functions during training [9; 8]. A method addressing the inherent GAN training issues at the data level is lacking. Motivated by this, we develop a solution, Two-stage CGAN, to enhance the quality of samples from minority classes when training GAN models on imbalanced datasets.

The success of deep learning-based generative models in producing realistic images, videos, and audio has led to a critical consideration: how to effectively assess the quality of synthetic samples. The Fréchet Inception Distance (FID) is commonly used for evaluating images [1], but it may not be directly applicable to sequential data, such as text, audio, or time series, due to the absence of a widely accepted pre-trained model designed for extracting feature vectors from time series data. The demand for such a metric is even more necessary given the challenges in evaluating time series data through visual inspection compared to image data. Motivated by this, we introduced a comprehensive metric called Fréchet Fourier-transform Auto-encoder Distance (FFAD) [10], to assess the quality and diversity of synthetic time series samples and provide an avenue for refining generative models for time series generation.

1.2 Challenges

Class-imbalance presents significant challenges across various domains. In cyber attack detection, the targeted nature of attacks and the evolving tactics of cybercriminals result in a scarcity of valid observations for training machine learning classifiers [11]. For instance, Denial of Service (DOS) attacks may constitute only 0.27% of training and 0.14% of test-

ing datasets [12]. Similarly, in Medicare fraud detection, a mere 25 fraudulent transactions among 1,000,000 normal ones yield a striking imbalance ratio of 0.06%. The medical field also grapples with extreme class-imbalance, exemplified by cancer diagnosis scenarios where 97% of patients are non-cancerous, underscoring the need for robust classifiers capable of handling such skewed distributions [13]. This issue extends beyond cybersecurity and healthcare, impacting diverse areas such as information retrieval systems, oil spill detection in radar images, land use classification in remote sensing [14], space weather forecasting [7], and other natural science domains.

Numerous solutions have been proposed to address the class-imbalance issue, with cost-sensitive methods being a traditional approach. However, these methods face significant challenges. Deriving an accurate cost matrix is often difficult, as actual values are rarely available from data or expert sources. This limitation is compounded by the impracticality of manually testing various cost factors, leading to increased learning expenses [15; 16]. Real-world applications often involve multidimensional costs, such as monetary and reputational factors, which are frequently overlooked in machine learning literature [17]. Moreover, most machine learning algorithms are not inherently designed for cost-sensitive learning, necessitating time-consuming and algorithm-specific modifications to effectively utilize cost matrices [18]. While ensemble learning methods are commonly employed in cost-sensitive learning, they come with their own drawbacks. These include increased training time and a higher risk of overfitting, which must be carefully considered when implementing such solutions [19].

Beyond algorithm-level solutions, numerous data-level remedies have been proposed to address class-imbalance issues. Authors of [5] explored various undersampling, oversampling, and mix-sampling techniques for constructing training datasets from imbalanced data. While these sampling-based methods are efficient, they offer limited improvements as they neither introduce nor utilize new data. An alternative approach involves generating synthetic samples from existing data. This can be categorized into statistic-based and generative model-based methods. SMOTE [20], a representative statistic-based method, creates new samples between minority instances and their nearest neighbors of the same class. However, it's crucial to note that such methods: (1) generate point-in-time data rather than time series data; and (2) assume normality, which may not hold in many real-world applications. In our study, we observe that peak X-ray fluxes of solar flares follow a power-law distribution, influencing the distribution of magnetic-field physical parameters in SWAN-SF. From a novel perspective, we focus on generative-based methods, particularly Generative Adversarial Networks (GANs). GANs offer a promising solution to class-imbalance by learning the underlying distribution of existing data and synthesizing new samples based on this distribution. This approach facilitates the creation of a more balanced training dataset, thereby enhancing the performance of multivariate time series classification tasks.

Furthermore, we aim to address several key challenges in this work, including: (1) determining model training progress and optimal model selection; (2) assessing the realism of synthetic data and its effectiveness in solar flare forecasting; (3) examining the impact of class imbalance on the quality and diversity of minority class synthetic samples generated

by CGAN models; and (4) developing an effective evaluation metric for assessing the realism of generated time series, analogous to the Fréchet Inception Distance (FID) used in image evaluation, with the potential to refine time series-based generative models.

1.3 Outline

The remainder of this dissertation is structured as follows. Chapter 2 presents background information on class-imbalance remedies, Generative Adversarial Networks (GAN), time series generation using GAN, the SWAN-SF dataset, and evaluation metrics for assessing synthetic data. This will provide definitions for class-imbalance issue and generative adversarial network, as well as other relevant details about datasets and evaluation metrics used throughout the text.

In Chapter 3, our work on the generation of synthetic multivariate time series is described. This section begins with a discussion of the conditional GAN, the data-generation approach. Secondly, we describe two ways to verifying the resemblance between real and synthetic data based on their descriptive statistics. In addition, we designed a series of experiments to evaluate the efficacy of the generated samples as a class-imbalance remedy for solar flare forecasting.

In Chapter 4, we evaluate the impact of the class-imbalance problem on training CGAN models. Show the ineffectiveness of common remedies for training GANs on imbalanced datasets, such as oversampling and undersampling. Additionally, we introduce a novel framework, the Two-stage CGAN, designed to progressively improve the quality of minority class

samples in both image and time series contexts.

In Chapter 5, we introduce the Fréchet Fourier-transform Auto-encoder Distance (FFAD), a novel metric that utilizes Fourier transform and Auto-encoder techniques to evaluate the quality and diversity of generated time series samples.

Finally, in Chapter 6, we will provide a summary of our findings and discuss future work.

CHAPTER 2
BACKGROUND

2.1 Class-imbalance Issue and Remedies

One challenge present in many, if not all, natural hazard forecasting problems is the issue of class-imbalance, and flare forecasting is no exception to this issue. Class imbalance typically occurs when there are more instances of some classes than others. It is common to use special remedies to address the class imbalance if it is present, since standard classifiers can be overwhelmed by the majority classes and neglect the minority ones. In typical class-imbalance situations, the minority class is the class of interest and therefore cannot be ignored. As a result, two approaches to overcoming the imbalance issue are established: either reduce the class skew at the data level or alternate the learning procedure at the algorithm level.

2.1.1 Data-/Algorithm-Level Remedies

As the representative method of data level, resampling is a classifier-independent technique for addressing imbalanced data, and it is accomplished in one of three ways: (1) Oversampling: selecting and duplicating samples of the minority class; (2) Undersampling: removing samples of the majority class; or (3) Hybrid: coupling the oversampling and undersampling methods when multi-class data are present [7]. The authors of [21] show that the classification performance improves when the above class-imbalance remedies are applied to a solar flare benchmark dataset, namely SWAN-SF [22]. However, random undersampling can jeop-

ardize the preservation of important concepts because it removes the most samples from the majority classes [23]. Random oversampling is susceptible to the risk of overfitting because it neither introduces nor utilizes new data. To reduce such risks in the image domain, we can perform transformation-based data augmentation, a heuristic oversampling strategy for dealing with the lack of data. To achieve this, the current examples are subjected to one or more data transformations, such as random rotation, translation, reflection, cropping, blurring, sharpening, and hue adjustment. These transformations are not applicable in all circumstances. A reflection or affine transformation, for instance, would alter the chirality of a picture of a solar filament. In addition, it is challenging to apply transformation-based data augmentation to feature-based data points or sequential data such as time series and text data [24]. To deal with such a situation, SMOTE [20], a heuristic oversampling method, is introduced by constructing new synthetic samples between minority instances and their nearest neighbors of the same class. In [20], Synthetic Minority Oversampling Technique, namely SMOTE, was introduced to create new samples between minority instances and their nearest neighbors of the same class. In addition, authors of [2] devised three naïve random synthetic oversampling methods, namely Random Uniform Synthetic Oversampling (RUSO), Random Normal Synthetic Oversampling (RNSO), and Random NOise Synthetic Oversampling (RNOSO), were employed to generate synthetic samples of the minority classes in the flare forecasting problem. RUSO generates new samples from the uniform distribution between the minimum and maximum values of each feature of minority samples in the training set. In contrast, RNSO uses the normal distribution with the mean and standard deviation

values of each feature of minority samples in the training set. RNOSO generates new samples by sampling noise terms from the normal distribution with mean of 0 and standard deviation of 0.1, and adds the noise to the existing minority samples. Although [2] demonstrates the benefit of using various statistic-based synthetic sampling methods over naïvely oversampling and undersampling, it is important to note that: (1) they do not generate time series data, however, the development of recurrent neural networks and generative modeling opens the door to generating sequential data, such as videos and music, and time series [25; 26]; (2) they work under the assumption of normality, which is not necessarily valid in many real-world applications. In our study, we know that the peak X-ray fluxes of solar flares follows a power-law distribution, and the distribution of each of the magnetic-field physical parameters of SWAN-SF is impacted by that. In addition, the statistic-based oversampling method may suffer when the separation between majority and minority clusters is not always obvious, resulting in noisy samples [23]. Moreover, the method is based on information from the local area, not the overall distribution of minority classes [14]. Generative Adversarial Networks provide an alternative method for addressing the lack of data by learning the underlying distribution of real samples and then generating new realistic samples [27; 28].

In contrast to data-level solutions, algorithm-level approaches are directly implemented within the training procedures of classifiers. These approaches can be categorized into three main types: (1) Classifier adaptation: This involves adapting existing machine learning algorithms to a particular imbalanced dataset [29]. While effective, this method is time-consuming and requires a deep understanding of the classifier; (2) Ensemble learning: This

approach combines several base models to construct an optimal predictive model. For example, the majority class samples can be divided into multiple small portions balanced with minority classes, training multiple individual classifiers and yielding the final decision through a voting mechanism [30]. However, ensemble modeling has drawbacks, including increased training time and a higher risk of overfitting [19]; (3) Cost-sensitive learning: This method designates a high misclassification cost to minority classes with the objective of minimizing the total cost [16]. However, deriving an accurate cost matrix is often challenging, as actual values are rarely available from data or expert sources [15; 16]. This limitation is compounded by the impracticality of manually testing various cost factors, leading to increased learning expenses. Moreover, real-world applications often involve multidimensional costs, such as monetary and reputational factors, which are frequently overlooked in machine learning literature [17]. Overall, data-level solutions, which involve manipulating existing data or generating synthetic samples to achieve a balanced class distribution, offer greater versatility in addressing class imbalance issues compared to algorithm-level approaches. Therefore, this study primarily focuses on these data-level techniques as a means of mitigating class imbalance.

2.1.2 Impacts of Class Imbalance on CGAN Training

The class imbalance issue can also have an impact on the CGAN training procedure. For example, the authors of [8] suggest that standard GANs cannot be used to produce minority-class images from an imbalanced dataset. There have been few research addressing this imbalance issue. BAGAN [9] is an enhancement tool for producing high-quality photographs

of minority groups by accomplishing the following: (1) Using an autoencoder to initiate the GAN training, allowing the model to learn accurate class-conditioning information in the latent space. (2) Combining the real/fake and classification losses at the discriminator into a single output. Using BAGAN as a foundation, the authors of [8] utilize supervised autoencoder and gradient penalty to overcome the instability problem when images from distinct classes appear similar. However, the previous researchers aim to address the imbalance problem at the algorithm level, either by applying autoencoder to discover latent features or by altering goal functions during the training stage. In Chapter 4, we investigate the issue of class imbalance in CGAN training, and and develop a data-level solution named Two-stage CGAN.

2.2 Generative Adversarial Networks (GANs)

Generative Adversarial Network is an emerging technique for modeling high-dimensional distributions of real samples implicitly [31]. Initially proposed in [27], the GAN learns to produce realistic data by training two components, the generator and the discriminator, in an adversarial manner. First, the generator is used to capture the data distribution by sampling random vectors from a latent space as inputs, and to produce samples similar to the real data. Next, the discriminator receives both generated samples and real samples as inputs, and estimates the probability of the input coming from the real data space. By training the generator and the discriminator simultaneously, a generator is enabled to gradually generate more realistic samples under the supervision of the real samples. This

process is repeated until the discriminator cannot distinguish the generated samples from the real ones. Typically, either the generator or the discriminator can be implemented by arbitrary multilayer neural networks consisting of fully connected networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs), depending on the nature of the data source.

The vanilla GAN has exhibited some limitations on the stability of the model training and the diversity of the generated sample [32]. Therefore, several works have investigated designing new architectures in order to mitigate the training issues, and improve the quality of the generated samples. For example, the Deep Convolutional GAN (DCGAN) utilizes convolutional neural networks as the generator and discriminator, and replaces pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator) to improve the training stability [33]. The Wasserstein GAN [32] introduces the Earth-Mover distance to improve the learning stability and provide a meaningful learning curve for tuning hyperparameters. The Info GAN [34] incorporates the representation learning by maximizing the mutual information between a fixed subset of the latent variables and the observations. The Variational GAN (VAEGAN) [35] combines Autoencoder and GAN to encode the real data as inputs of the generator instead of randomly sampling from a latent space, enabling the GAN model to achieve faster and more stable learning. The Conditional GAN (CGAN) [36] is another variant dedicated to improving the quality of the generated samples and controlling the classes of the synthetic samples by utilizing conditional information. The most common form of conditional information is the class labels. All the reviewed approaches and their

key features are organized in Table 2.1 for convenience.

Table 2.1 The table summarizes fourteen studies related to this work. The Over-/Under-sampling (Row 1) is the simplest one which is considered as our baseline model. Row 2 corresponds an effective method in multi-class problems that couples oversampling and undersampling by preserving the distribution of the subclasses in the original dataset. Rows 3 and 4 describe statistic-based methods. Rows 5-14 list several GAN-based methods. The type of the generated data for each method is specified in the 3rd column.

No.	Methods	Types	Features
1	Oversampling Undersampling	Time series	The simplest one; No new data introduced.
2	Advanced Oversampling [7]	Time series	Applied on multi-class problems.
3	SMOTE [20]	Point-in-time	Provides statistical interpretations.
4	RUSO/RNSO/RNOSO [2]	Point-in-time	Provides statistical interpretations.
5	Vanilla GAN [27]	Time series	Generating single-class samples.
6	WGAN [32]	Time series	Stable and faster training by using a meaningful objective function.
7	InfoGAN [34]	Time series	Learns interpretable latent variables in an unsupervised manner.
8	VAEGAN [35]	Time series	Stable and faster training.
9	CGAN [36]	Time series	Generates multi-classes samples by using conditional information; Stable and faster training.
10	C-RNN-GAN [26]	Time series	Generates single category musical data.
11	RCGAN [37]	Time series	Generates privacy-free medical data.
12	DoppelGANger [38]	Time series	High-fidelity; Privacy-free; Deal with mix-type data.
13	[39]	Time series	Learns the conditional probability distribution of features by GAN.
14	TimeGAN [40]	Time series	Incorporates conditional temporal dynamics into the unsupervised GAN.

Numerous GAN applications have been proposed to deal with different demands, from art, science, finance, drug discovery to video games, and have achieved great success. In the computer vision domain, synthetic image generation has been tested in scenarios, such as cartoon characters [41], face frontal views [42], and new human poses [43]. Image-to-image translation [44; 45] and text-image synthesis [46] applications enable users to transfer objects between different styles or different formats. Moreover, image super-resolution [47; 48] and motion stabilization [49] applications are especially helpful in autonomous driving and navigation tasks since object detection accuracy is improved by utilizing optimized images or videos. The last but not least avenue of applying GANs is data augmentation. Traditional data augmentation techniques usually perform a transformation pipeline on the existing instances of data, and it involves one or more of data manipulations, to name a few, random rotation, translation, reflection, cropping, blurring, sharpening, and hue adjustment. However, these transformations are not applicable to all situations. For example, the chirality of an image of solar filament would be changed if a reflection or affine transformation is performed. GAN provides an alternative way to perform the data augmentation. That is, to learn an underlying distribution of real samples, and to produce new realistic samples based on the learned distribution.

2.2.1 Time Series Generation Using GAN

Various projects in different domains have emerged to shed light on generating time series data by utilizing the Generative Adversarial Network, as shown in Table 2.1. In [26], use of a C-RNN-GAN was proposed as a method to generate musical data. This method applied a

unidirectional Long Short-Term Memory (LSTM) as the generator and a bidirectional LSTM as the discriminator. In [37], RGAN was developed as a privacy-preserving method for generating synthetic medical data in an effort to mitigate the concern regarding the utilization of the privacy-sensitive patient data to train machine learning models. DoppelGANger [38] is another framework designed for generating synthetic time series data with high-fidelity and sharing data with privacy-free properties. Particularly, it deals with mix-type datasets which contain continuous and discrete features. In [50], GAN was utilized as a data augmentation method for generating synthetic biosignal data, including electroencephalographic (EEG) and electrocardiography (ECG). The improved Wasserstein GAN was employed to generate synthetic spiking time series in the banking domain [51]. In [39], the authors used GAN to learn the conditional probability distribution of the key features to generate synthetic time series data. TimeGAN [40] combined the versatility of the unsupervised GAN approach with the control over conditional temporal dynamics. This method has two more autoencoding components, including an embedding function and a reconstruction function trained jointly with the generator and the discriminator components. This structure enables the model to iteratively learn to encode features, generate representations, and adjust weighting parameters according to the objective function.

2.3 Evaluation Metrics for Generative Models

Deep learning has shown remarkable success in numerous tasks and domains, highlighting its effectiveness in tackling complex challenges. Notably, it particularly excels in generative

models such as Generative Adversarial Networks (GANs) [27], Conditional GANs (CGANs) [36], and Variational Auto-encoders (VAEs) [52], showcasing their capacity to produce realistic images, artwork, and time series data [33; 45; 53].

Automatic evaluation metrics such as Inception Score (IS) and Fréchet Inception Distance (FID) have been developed to assess the quality and diversity of generative samples [1; 54]. IS uses a pre-trained image classification network, Inception v3 [55], to evaluate the conditional and marginal distributions of synthetic samples based solely on the generated data. In contrast, FID measures the Fréchet distance between real and generated images, employing feature vectors extracted from the Inception v3 model. FID has emerged as the preferred metric (as depicted in Fig. 2.1) due to its ability to effectively capture the level of discrepancy through its monotonically increasing behavior, whereas Inception Score may fluctuate, remain constant, or even decrease in the worst cases [1]. However, FID is only employed for the evaluation of images [56], and may not be as directly applicable to sequential data, such as text, audio, or time series, due to the absence of a widely accepted pre-trained model designed for extracting feature vectors from time series data.

The Fréchet distance provides a way for measuring the similarity between curves [57]. Introduced in 2017, the Fréchet Inception Distance (FID) score is the current standard metric for evaluating the quality of generative models in image generation. Using the feature vectors derived from the Inception v3 model [55], FID calculates the distance between real and generated images. Specifically, the final pooling layer preceding the classification of output images is used to capture computer-vision-specific features of an input image. In practice,

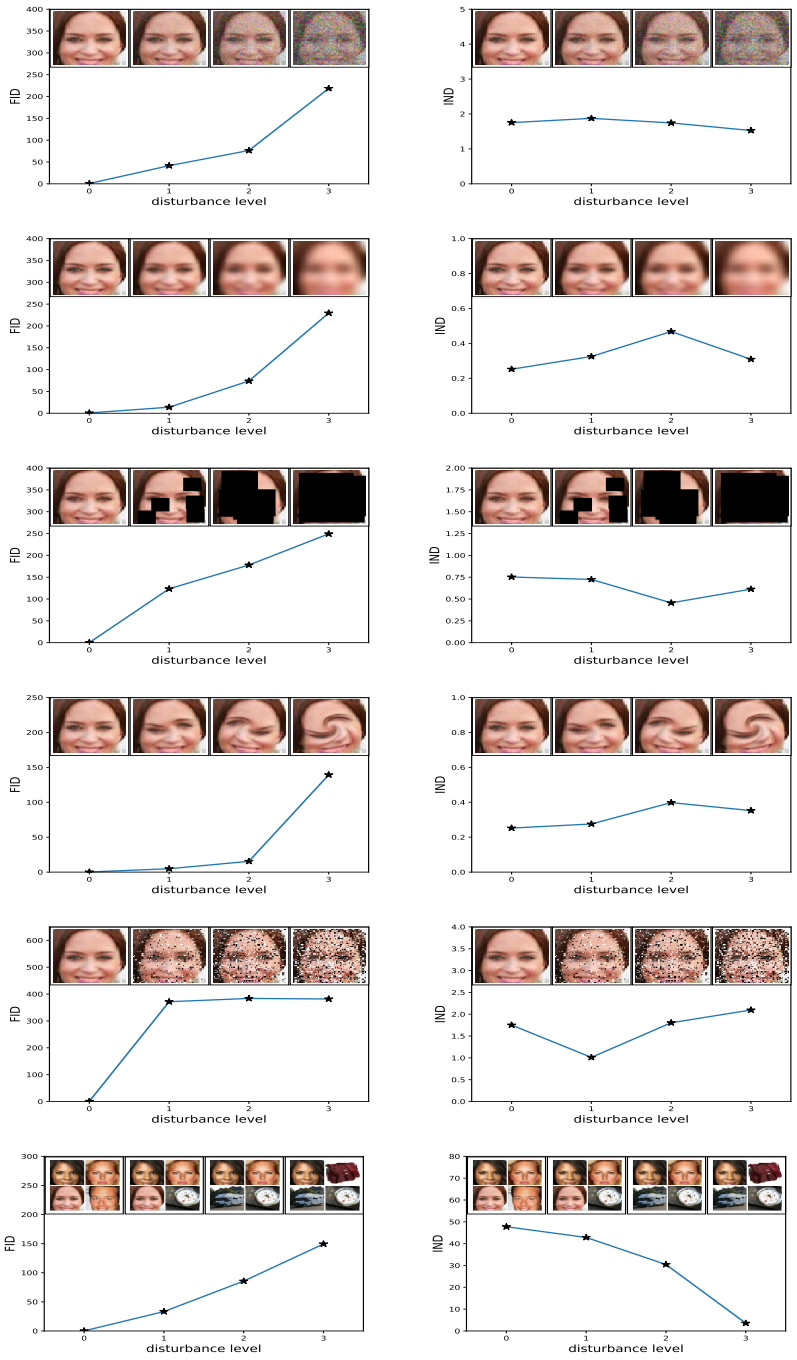

Figure 2.1 FID and Inception Score Comparison]**Left:** FID and **right:** Inception Score are evaluated for **first row:** Gaussian noise, **second row:** Gaussian blur, **third row:** implanted black rectangles, **fourth row:** swirled images, **fifth row.** salt and pepper noise, and **sixth row:** the CelebA dataset contaminated by ImageNet images. Left is the smallest disturbance level of zero, which increases to the highest level at right. The FID captures the disturbance level very well by monotonically increasing whereas the Inception Score fluctuates, stays flat or even, in the worst case, decreases [1].

each input image is represented as a feature vector. X and Y are feature vectors of the real and synthetic samples. Then, multivariate FID can be computed based on the formulation in Eq. 2.1 [58]. μ_X and μ_Y are the vector magnitudes X and Y, respectively. $Tr(.)$ is the trace of the matrix, while Σ_X and Σ_Y are the covariance matrices of X and Y. Lower FID values indicate higher quality and diversity in synthetic samples.

$$\text{Score} = ||\mu_X - \mu_Y||^2 + \text{Tr}(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y}) \tag{2.1}$$

Moving beyond image generating models, the authors of [59] introduced the Fréchet ChemNet Distance (FCD) as a evaluation metric for generative models in the context of molecular structures relevant to drug discovery. Diverging from the FID, FCD derives feature vector representations for each molecules by utilizing the penultimate layer of the ChemNet [60]. Subsequently, the Fréchet Distance is computed based on the distribution of real and generative samples.

2.4 Datasets

This work primarily utilizes three data sources, including MNIST, UCR, and SWAN-SF datasets, explored across Sections 3, 4, and 5. Each dataset plays a pivotal role as a foundational component for exploring and validating a wide range of methodologies and models specific to their respective chapters.

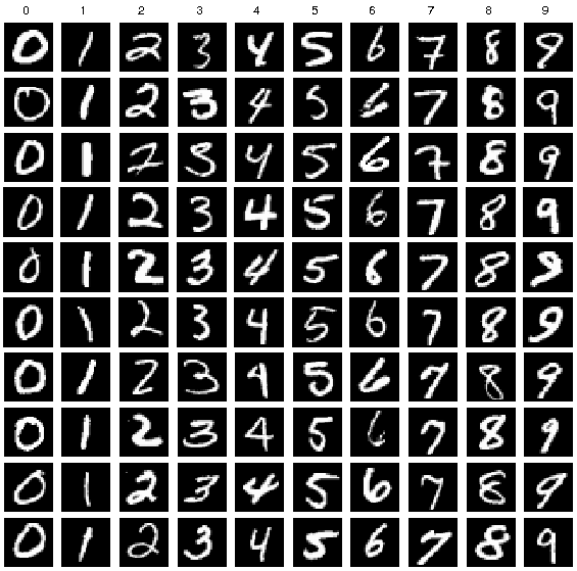

Figure 2.2 The plot illustrates handwritten digit examples from the MNIST dataset, with each grayscale image measuring 28x28 pixels. The dataset comprises 10 classes, representing the digits 0 through 9.

2.4.1 MNIST Dataset

MNIST is a benchmark database of handwritten digits that is frequently used to train and evaluate machine learning algorithms [61]. The original dataset consists of 10 classes, which are distributed evenly across 60,000 training images and 10,000 testing images. Each image being grayscale and normalized to fit into a 28x28 pixel box, as shown in Fig. 2.2. MNIST has been pivotal in advancing the development of various machine learning techniques, serving as a standard testbed for evaluating the performance of classifiers, neural networks, and deep learning models.

2.4.2 UCR Dataset

The UCR Time Series Classification/Clustering Repository, curated by the University of California, Riverside (UCR), stands as a prominent resource in the fields of time series analysis and data mining [62]. It plays a pivotal role as a benchmark for the development and assessment of algorithms and models tailored for time series classification, clustering, and related tasks. The UCR datasets consist of 117 datasets of fixed length. In Chapter 5, we specifically utilized 97 out of the 117 datasets, as our Fourier transform-based methodology was found to be incompatible with the remaining 20 datasets.

2.4.3 SWAN-SF Benchmark Dataset

Solar flares are understood to be dynamical phenomenon which have clear pre-flare and post-flare phases [63]. However, many studies utilize point-in-time measurements, e.g., [64; 65; 66], to predict flares' behavior in the future. To expand the horizon of flare forecasting, a recent study generated a large collection of multivariate time series data of flares' magnetic fields extracted from solar photospheric vector magnetograms in Space weather HMI Active Region Patch (SHARP) series [67]. The benchmark dataset, named Space Weather ANalytics for Solar Flares (SWAN-SF), is meant to serve as a testbed for flare forecasting models [22]. It is hoped that using this dataset some significant improvements in the performance and robustness of the forecasting models be achieved. This benchmark dataset is made openly available on Harvard Dataverse repository [68]. SWAN-SF has 5 classes, including four flare classes of X, M, C, and B, with an additional class labeled as NF representing absence of

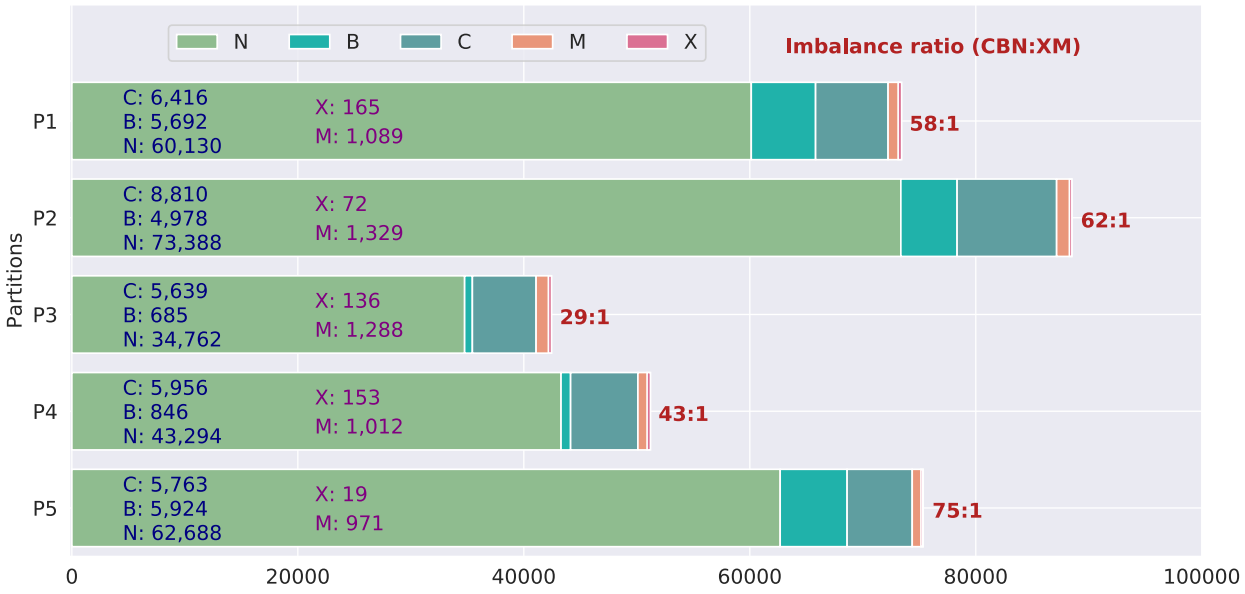

Figure 2.3 The plot illustrates the distribution of the 5 flare classes in SWAN-SF dataset. The flare counts and the imbalance ratio (font in red) per partition are annotated. In this study, the flare instances of X and M classes make up the positive class, and the C, B, and N classes account for the negative class.

any of the listed flares. Each multivariate time series is labeled by looking at the strongest flare event recorded in the 24-hour prediction window. This interval follows the 12-hour observation window from which the magnetic-field parameters are calculated. In this study, we simplify the task to a binary classification by merging the stronger instances (i.e., X- and M-class flares) to form the positive class, and the weaker instances (i.e., of C, B, and NF classes) to represent the negative class. The extreme class imbalance exhibited by SWAN-SF is illustrated in Fig. 2.3, with each class's sample size annotated. A proper treatment of this imbalance is the objective of this study.

The SWAN-SF is made up of five temporally non-overlapping partitions covering the period from May 2010 through August 2018. Each partition contains approximately an equal

number of X- and M-class flares, and there are a total of 6,234 flare records and 324,952 non-flaring records. Each flare record is a multivariate time series (MVTS) with 60 time steps, each of which has 51 magnetic field parameters (for the definition of the parameters see Table 1 in [22]). We limit our investigation to only four of these 51 parameters, abbreviated to TOTUSJH, ABSNJZH, SAVNCPP, and TOTBSQ, which have been listed as the most relevant to the flare forecasting in several studies including [67] and more recently in [69]. Moreover, based on how they are calculated it is easy to see that many of these parameters are highly correlated with each other and a small subset of them suffices our objective in this study.

One major concern for evaluating flare forecasting models is to determine evaluation metrics appropriate for the above-mentioned class imbalance. Many well-known performance metrics are significantly impacted by class imbalance [70], including accuracy, precision, and F1-score, which ignore the number of misclassified instances. From years of exploration, domain experts have agreed on two effective metrics, namely the *true skill statistic* (TSS) [71] and the updated *Heidke skill score* ($HSS2$) [72], as shown in Eq. 2.2 and Eq. 2.3, respectively. These are functions of the confusion matrix whose entries are true positive (tp), true negative (tn), false positive (fp), and false negative (fn). We will use both of these metricsf to evaluate the performance of our flare forecasting models.

$$TSS = \frac{tp}{tp+fn} - \frac{fp}{fp+tn} \tag{2.2}$$

$$HSS2 = \frac{2\cdot((tp\cdot tn)-(fn\cdot fp))}{(tp+fn)\cdot(fn+tn)+(fp+tn)\cdot(tp+fp)} \tag{2.3}$$

CHAPTER 3
SYNTHETIC MULTIVARIATE TIME SERIES GENERATION

Our primary objective is to evaluate the effectiveness of Conditional GAN as a potential solution to the class-imbalance issue in the SWAN-SF dataset. In this chapter, we focus on accurately assessing the contribution of CGAN-based synthetic multivariate time series for SWAN-SF and determining whether these generated time series are reliable for machine learning use.

3.1 Conditional GAN

The algorithm we employ is the Conditional Generative Adversarial Network (CGAN) whose architecture is illustrated in Fig. 3.1. Several reasons make us decide to utilize this algorithm: First, CGAN can control the category of generated samples, allowing us to generate samples of minority classes to mitigate the class-imbalance issue. Second, it can provide stable and faster training compared to the vanilla GAN as stated in [73]. Third, the category information of instances in the SWAN-SF dataset is available as conditional information for training CGAN models. We choose LSTM networks as the fundamental components in both the generator and the discriminator since our subject is sequential data.

As mentioned in Section 2.2, the ultimate goal of a generator (G) is to generate an output with similar characteristics as the real data. As seen in Fig. 3.1, the algorithm takes in a random input vector Z_n, which is a tensor with the shape of [$batch\_size$, $sequence\_length$, $latent\_dim$]. In our case, the shape is $[32, 60, 3]$ for 32 multivariate time series in a batch, each of length 60 and the latent dimension of 3. The conditional vector (C_n), as a type of

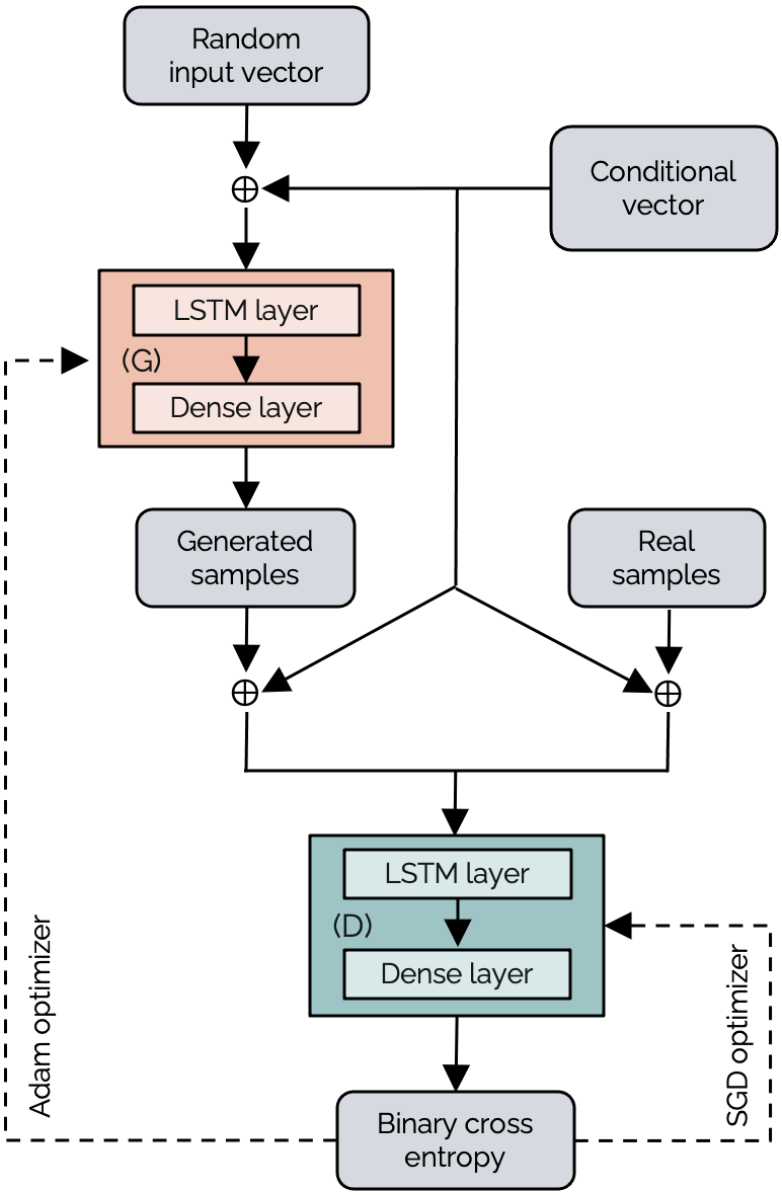

Figure 3.1 This is the framework of the CGAN algorithm, including components of the generator (G) and the discriminator (D). Each component is processed by the combination of the LSTM layer and the Dense layer. The inputs of the generator are random input vectors concatenated with conditional vectors. The inputs of the discriminator are either generated or real multivariate time series with conditional vectors. The binary cross-entropy is the criterion for optimizing the model.

auxiliary information, has the shape of $[32, 60, 2]$ since the binary labels are encoded into a one-hot representation. By concatenating Z_n and C_n, we obtain a tensor of shape $[32, 60, 5]$ as the final input of the generator. Note that the latent space dimension, as a hyperparameter, is determined by the dimension of the parameters and the conditional information. We empirically assume that the total dimension of the latent space and the conditional information should be similar to the dimension of the parameters being produced. Besides, the dimension of the latent space and the conditional information should be balanced, which means neither should dominate the inputs of a generator. The outputs of the generator, regarded as the generated or synthetic samples, are calculated by going through the LSTM and Dense layers pipeline. The LSTM layer controls the memorizing process using a gating mechanism. Meanwhile, the Dense layers guarantee that the generated samples can maintain the same shape as the real data, i.e., $[32, 60, 4]$ where 4 stands for four magnetic field parameters mentioned in Section 2.4.

The task of a discriminator (D) is to classify inputs as either being the real or generated samples produced by the generator. As Fig. 3.1 illustrates, the discriminator takes both the real and the generated multivariate time series samples as the inputs. To simplify the notation, we use $\widetilde{X_n}$ to denote either real (X_n) or synthetic samples ($G(Z_n|C_n)$) when the difference is clear from the context. By feeding C_n into D, the discriminator produces judgments about whether the sample is generated or real and evaluates if the category of the generated sample corresponds to its conditional information. Finally, the binary cross-entropy loss calculated between the predicted and the ground truth values is used to update

the weighting parameters of the generator and the discriminator using the backpropagation.

So far, we have described the structures and functionalities of the generator and the discriminator. Next, we define the objective function used for optimizing the algorithm. In our framework, the objective function is divided into two parts: the generator loss ($Loss_G$) and the discriminator loss ($Loss_D$). The discriminator loss is obtained by calculating the cross-entropy between the ground-truth and the outputs of the discriminator, as shown in Eq. 3.1,

$$Loss_D(\widetilde{X_n}|C_n, y_n) = -\mathrm{CE}\Big(D(\widetilde{X_n}|C_n), y_n\Big) \tag{3.1}$$

where $\widetilde{X_n}$ is the set of inputs of the discriminator, and C_n is the conditional vector. $D(\widetilde{X_n}|C_n)$ returns the likelihood of $\widetilde{X_n}$ being a real or a generated sample, and CE stands for the cross-entropy loss function. Note that $\widetilde{X_n}$ is composed of two different types of data sources, as formulated in Eq. 3.2.

$$\widetilde{X_n} = \begin{cases} X_n & \textbf{if } \textit{inputs are real samples} \\ \mathrm{G}(Z_n|C_n) & \textbf{if } \textit{inputs are generated samples} \end{cases} \tag{3.2}$$

Correspondingly, y_n takes two different values depending on the source of the sample in $\widetilde{X_n}$.

$$y_n = \begin{cases} \boldsymbol{1} & \textbf{if } \textit{inputs are real samples} \\ \boldsymbol{0} & \textbf{if } \textit{inputs are generated samples} \end{cases} \tag{3.3}$$

The generator loss ($Loss_G$) is also formulated in Eq. 3.4, where the input $G(Z_n|C_n)$ is the generated samples, and its corresponding predictions are $D\Big(G(Z_n|C_n)|C_n\Big)$. To optimize the generator, we need to guide the discriminator to classify the generated samples as real. To do so, we initialize the ground-truth labels with $\mathbf{1}s$ (same as the real samples). By minimizing $Loss_G$, the predictions of the discriminator approach $\mathbf{1}s$ gradually, indicating the generated samples are realistic-enough that the discriminator cannot distinguish them from the real samples.

$$Loss_G(Z_n|C_n) = -\mathrm{CE}\Bigg(D\Big(G(Z_n|C_n)|C_n\Big), \mathbf{1}\Bigg) \tag{3.4}$$

3.2 Methodology

There are two main concerns in evaluation of GAN models and their synthetically generated data: (1) to determine the learning progress, and (2) to examine the effectiveness of synthetic data for the original problem. Regarding the former, in most image-based GAN projects, researchers can determine the training progress by visually examining the synthetic images. However, the visual inspection of synthetic time series does not give us much evidence as to whether the synthetic samples are realistic or not. To address this concern, we present two statistical-based approaches to handle the model selection issue. Regarding the latter, we design multiple experiments by applying different class-imbalance remedies to tackle the flare forecasting problem. We elaborate on our methodologies in the following text.

3.2.1 Model Selection Using Distributions of Statistical Features

To provide a statistical evaluation for our model, we compare a few descriptive statistics extracted from the real and synthetic time series data. This establishes a high-level similarity criterion that must be satisfied if the distributions of the real and generated time series are indeed similar.

Suppose we have sets of real (T) and synthetic (S) samples, with equal number of multivariate time series. For each instance, we extract its mean, median, and standard deviation. We then construct the corresponding probability distributions P_T and P_S, with setting the bin size to M. To quantitatively measure the similarity, we calculate the Kullback–Leibler (KL) divergence [74] between distributions of P_T and P_S using Eq. 3.5. The KL-divergence is a non-negative measure, which means $D_{KL}(P_T||P_S) \geq 0$. The smaller value indicates the higher similarity between P_T and P_S.

$$D_{KL}(P_T||P_S) = \sum_{m \in M} P_T(m) \cdot log\Big(\frac{P_T(m)}{P_S(m)}\Big) \tag{3.5}$$

3.2.2 Model Selection Using Adversarial Accuracy

The Adversarial Accuracy, as formulated in Eq. 3.6, is put forward by Yale et al. [75], which is used for measuring the similarity of two sets of data samples through their nearest neighbors.

$$AA_{TS} = \frac{1}{2}(\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(d_{TS}(i) > d_{TT}(i)) + \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(d_{ST}(i) > d_{SS}(i))) \tag{3.6}$$

$$\begin{cases} d_{TS}(i) = min_j ||X_T^i - X_S^j||_2 \\ d_{TT}(i) = min_{j,j\neq i} ||X_T^i - X_T^j||_2 \end{cases} \tag{3.7}$$

In Eq. 3.6, the subscripts T and S refer to the sets of real and synthetic samples, respectively. The distance function d is defined in Eq. 3.7 as the minimum (Euclidean) distance between each real sample X_T^i and all synthetic samples X_S^j (i.e., $d_{TS}(i)$), and all other real samples X_T^j (i.e., $d_{TT}(i)$). The shortest distance generally means the highest similarity between two samples. If $d_{TS}(i) > d_{TT}(i)$, it means no synthetic sample is found in S that is more similar to X_T^i than any other real samples in T. Otherwise, a synthetic sample, which is more similar to the X_T^i, can be found. A realistic sample is generated when $d_{TS}(i) < d_{TT}(i)$. The range of Adversarial Accuracy is $[0, 1]$. The outcome 1 indicates that there is no resemblance between the set of real samples and the set of synthetic samples. The outcome 0 indicates that the two sets are exactly the same, yielding no new information. The desirable outcome of Adversarial Accuracy is close to 0.5, implying that the real and synthetic samples generated by the generators are indistinguishable [75].

3.2.3 Synthetic Data v.s. Over-/Under-Sampling

To assess the effectiveness of the synthetic data, we design several experiments where we compare the impact of different balancing remedies on the classification of flaring and non-flaring instances of SWAN-SF, with that of balancing using our synthetic data. As shown in Table 3.1, we set up three groups of experiments, namely A, B, and C, and each comprises

four experiments. For A and B, the primary difference between them is that in the former we utilize the *last-value* statistic of MVTS samples, whereas in the latter, median and standard deviation of time series are used. The *last-value* is literally the last value of each time series. This makes our results comparable with those in [2] where point-in-time data were used. The mean statistic is sensitive to outliers, which makes us eliminate it for flare forecasting. For C, we aim to examine the effectiveness of synthetic samples in their original high-dimensional format, i.e., *time series*. The question is whether the unwanted noise of the synthetic MVTS samples was obscured by the summary descriptive statistics. Therefore, we conduct the experiments in C to verify the hypothesis.

For each group, we train four classifiers with the same parameter setting, but with different training datasets. The models in A1, B1 and C1 are trained on the highly imbalanced, real dataset without any changes. The models in A2, B2 and C2 are trained on the dataset that is made balanced by adding synthetic minority (flaring) samples. The models in A3, B3 and C3 are trained on the dataset that is made balanced by random oversampling of (i.e., duplicating) the minority instances. Lastly, the models in A4, B4 and C4 are trained on the dataset that is made balanced by random undersampling of the majority (non-flaring) instances. The models in A1, B1 and C1 are considered as the baseline.

3.3 Experiments

In this section, we conduct experiments to evaluate the effectiveness of the proposed assessment methods. First, we show the results of two model selection methods based on the

Table 3.1 The table lists all experiments carried out to examine various class-imbalance remedies. Groups of A and B have experimented on the extracted descriptive statistics of MVTS data. Group A utilizes the last value statistic of MVTS samples as inputs, whereas in group B, median and standard deviation of samples are used. All experiments in A and B utilize Partition 1 (P1) as the training set and Partitions 2, 3, and 5 as the test sets. Partition 4 is not involved in this experiment. The experiments in C are conducted to examine various class-imbalance remedies by taking time series as inputs. Similarly, Partition 1 is utilized as the training set and Partitions 2, 3, and 5 as the test sets. Partition 4 is reserved for validation of the hyperparameters.

Group	No.	Method	Description	Statistic
A	A1	Baseline (BL)	No data augmentation applied on P1.	last value
	A2	Synthetic Oversampling using CGAN (CGAN)	Adding synthetic flaring samples to the minority class of P1.	
	A3	Random Oversampling (RO)	Randomly oversampling samples of the minority class on P1.	
	A4	Random Undersampling (RU)	Randomly undersampling samples of the majority class on P1.	
B	B1	Baseline (BL)	No data augmentation applied on P1.	median & standard deviation
	B2	Synthetic Oversampling using CGAN (CGAN)	Adding synthetic flaring samples to the minority class of P1.	
	B3	Random Oversampling (RO)	Randomly oversampling samples of the minority class on P1.	
	B4	Random Undersampling (RU)	Randomly undersampling samples of the majority class on P1.	
C	C1	Baseline (BL)	No data augmentation applied on P1.	time series
	C2	Synthetic Oversampling using CGAN (CGAN)	Adding synthetic flaring samples to the minority class of P1.	
	C3	Random Oversampling (RO)	Randomly oversampling samples of the minority class on P1.	
	C4	Random Undersampling (RU)	Randomly undersampling samples of the majority class on P1.	

distributions of statistical features and the Adversarial Accuracy. Then, we present multiple experiments by applying different class-imbalance remedies to tackle the flare forecasting problem. Furthermore, we exhibit a quantitative analysis of the quality and usefulness of the synthetic flaring time series generated by the CGAN model to balance the training dataset.

3.3.1 Experimental Settings

After exploring various settings based on the defined objective function, we found that using the Adam Optimizer for the generator and the Gradient Descent Optimizer for the discriminator produced optimal results. We tune the performance of CGAN model by setting different hyper-parameters, i.e. latent space dimensions: $\{2, 3, 4, 5\}$, learning rates: $\{0.5, 0.1, 0.01, 0.001, 0.0001\}$, batch sizes: $\{16, 32, 64\}$. Empirically, we concluded our optimal hyper-parameter setting with the latent space dimension of 3, the conditional information dimension of 2 (since we have two classes), the learning rate of 0.1, the batch size of 32, and the LSTM hidden size of 100. The model was trained with 300 epochs, and intermediate models were saved at every five epochs. We have implemented CGAN using the TensorFlow 2.1 library [76].

For preprocessing of SWAN-SF, we linearly transformed all five partitions to the range $[-1, 1]$ for training the CGAN model and evaluations. We train the generator on Partition 1 of SWAN-SF, with the four magnetic field parameters mentioned in Section 2.4.

We employ the Support Vector Machine (SVM) as the standard classifier for experiment groups A and B in Section 3.2.3. The models are trained on Partition 1 and evaluated on Partitions 2, 3, and 5. Partition 4 is not involved in this experiment. For the experiments

in A, we use the same hyperparameters as were used in [2], i.e., kernel, C and gamma set to 'rbf', 0.5 and 8, respectively. Since the input of the experiments in group B has double dimensions compared to A (from 4 to 8), we adjust the hyperparameters accordingly following the instructions in [77; 78; 79], and set the kernel, C and gamma to 'rbf', 0.25 and 0.25, respectively. We conduct the time series-based classification experiments (group C in Section 3.2.3) using the time-series specific Support Vector Classifier (T-SVC). Similarly, Partition 1 is for training and Partitions 2, 3, and 5 are for evaluation. Partition 4 is reserved for validation of the hyperparameters. We performed a grid search on C and gamma to find the optimal setting, i.e. C: {0.001, 0.01, 0.1, 1, 10, 100}, gamma: {0.001, 0.01, 0.1, 1, 10, 100}, using the 'rbf' kernel. We conclude the optimal setting with C and gamma to 0.01 and 0.01.

3.3.2 Evaluation Using Distributions of Statistical Features

We have conducted this analysis on all of the four selected physical parameters, but for brevity, we present only the results for the physical parameter TOTUSJH. In Fig. 3.2, we compare the results by monitoring the improvement of the models at every 5 epochs, and the quality of the samples they generate. Specifically, we utilize $1,254$ real flare samples in Partition 1 of SWAN-SF and $1,254$ synthetic samples generated by the CGAN model in the evaluation. The columns A and B in Fig. 3.2 compare the distributions of the three descriptive statistics of the real and synthetic time series based on two intermediate models saved in the training process. Column A corresponds to a model trained after 50 epochs, whereas B shows the results after 250 epochs. Comparing A with B it is evident that, at

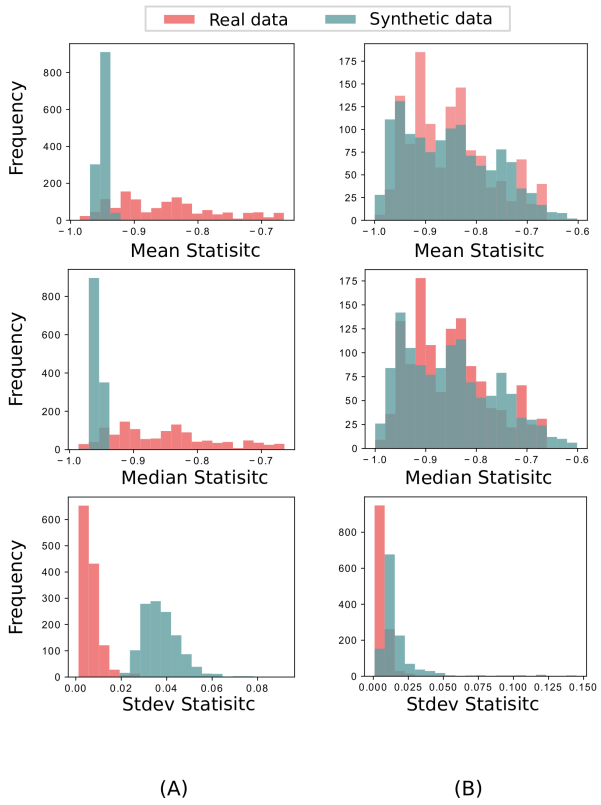

Figure 3.2 The plots show the distributions of mean, median and standard deviation of the physical parameter TOTUSJH and its synthetic counterpart using 20 equal-width bins. Columns A and B show the distributions of the descriptive statistics at two intermediate epochs, 50th and 250th, respectively.

least in terms of the three descriptive statistics, the generator gradually learn to generate synthetic time series which are more and more similar to the real flaring time series. To draw a more comprehensive picture, we calculate the Kullback–Leibler (KL) divergence between the distributions of three descriptive statistics of the real and the synthetic time series every 50 epochs. We observe, as shown in the Fig. 3.3, that the KL divergence decreases as training progresses. We found that on average, the models between the epochs 201-250 achieve the best performance, with lower KL-divergence for the mean, median, and standard

deviation distributions. We also see that the variance between the results produced by intermediate models trends downward until we surpass the 250 epoch mark. We further need to examine the overfitting issue. That is, the KL-divergence can be low if the CGAN model just memorizes the training set, resulting in no or limited new information produced. We assess this question in the next section.

3.3.3 Evaluation Using Adversarial Accuracy

As previously mentioned, the KL-divergence score can underestimate model performance if the model memorizes or replicates the training set. To address this, the Adversarial Accuracy method aims for a score close to 0.5, indicating that real and synthetic samples are indistinguishable. The evaluation of our intermediate models using Adversarial Accuracy is conducted for the physical parameter TOTUSJH, as an example. We again utilize $1,254$ real flare samples in Partition 1 of SWAN-SF and $1,254$ synthetic samples generated by the CGAN model. As the box plots suggest, the models between 201 to 250 epochs achieve the adversarial accuracy of 0.55, 0.60, and 0.68, in terms of mean, median, and standard deviation of the generated time series, respectively. This shows that the CGAN model can generate realistic synthetic samples by maintaining a good balance between underfitting and overfitting. Moreover, the Adversarial Accuracy results are consistent with our evaluation using KL-divergence.

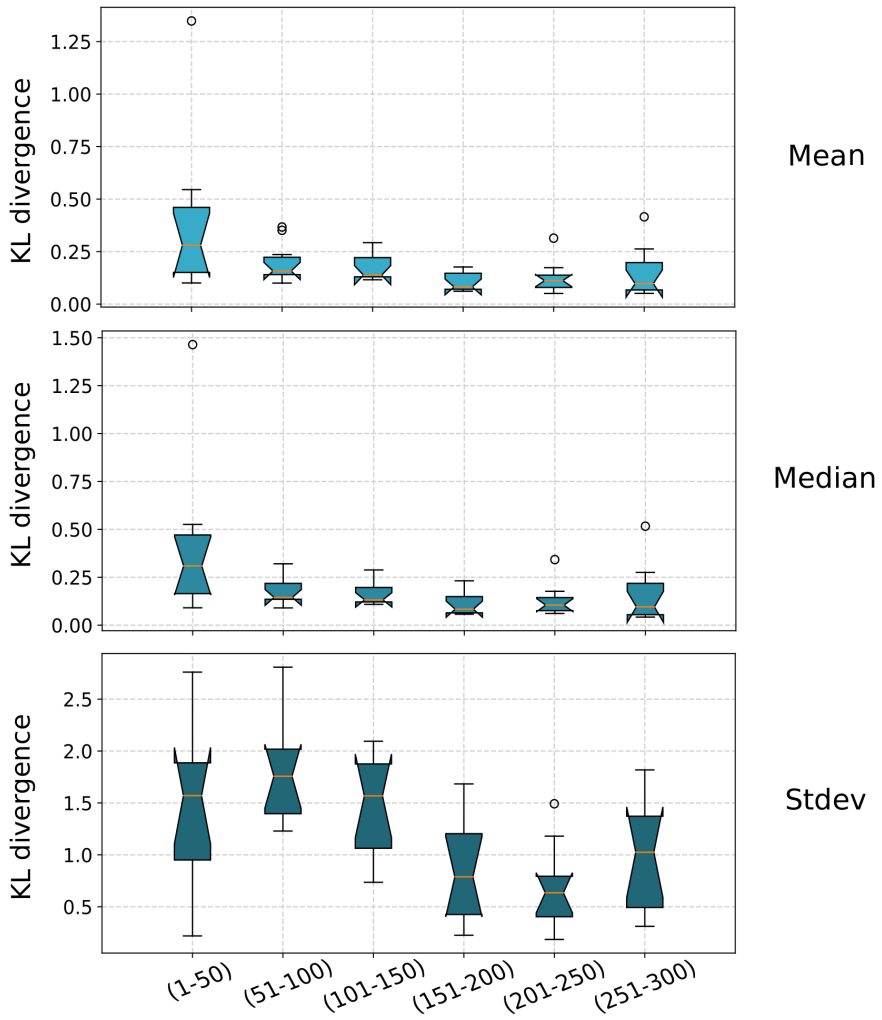

Figure 3.3 The plots shows the distributions of KL-divergence scores calculated by comparing distributions of synthetic samples and real samples across all intermediate models divided into six groups.

3.3.4 Examining Descriptive Statistics of Synthetic Time Series

We conducted two groups of flare forecasting-based experiments (A&B) to examine the effectiveness of the synthetic data using descriptive statistics. Four classifiers are trained for each group, with the same parameter setting but different training datasets. In A2 and B2,

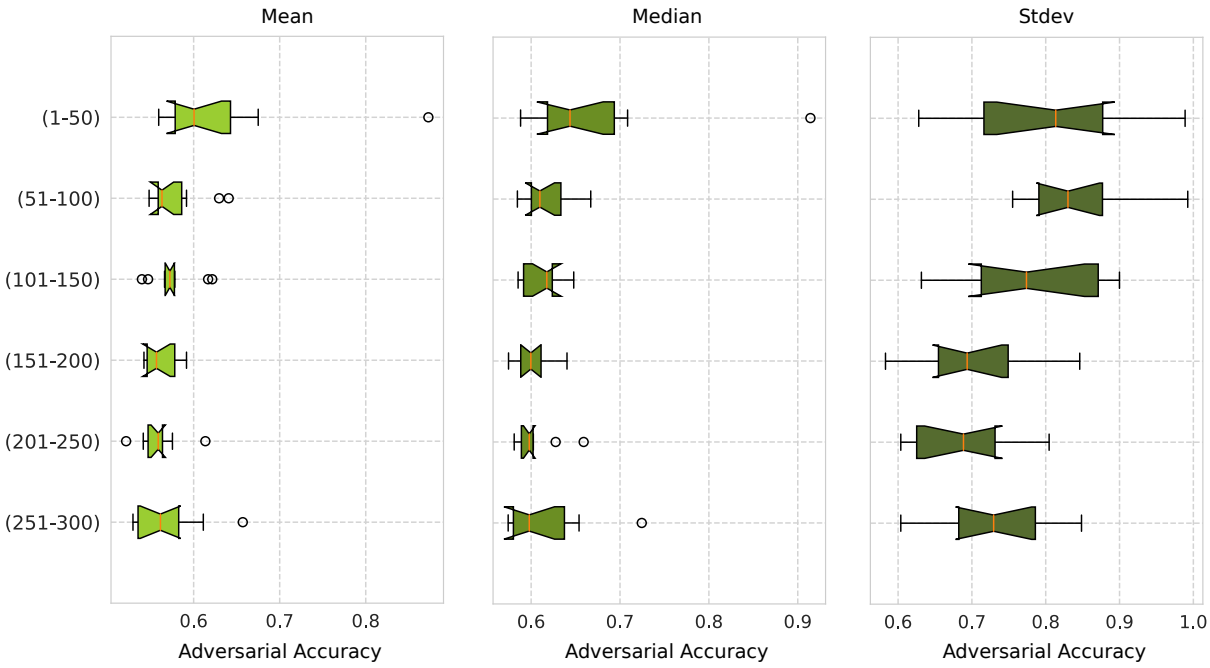

Figure 3.4 The box plots show the distributions of Adversarial Accuracy of the three descriptive statistics of TOTUSJH, namely mean, median, and standard deviation, evaluated with all intermediate models divided into six groups.

we generate $70,984$ synthetic flare samples to balance the training set. For A3 and B3, the training dataset is made balanced by random oversampling $70,984$ duplicates of the minority instances. For A4 and B4, the training dataset is made balanced by random undersampling $1,254$ of the majority instances. Of course, data manipulation is only served for the purpose of training, and test sets are made entirely of real data.

The results of the group A experiments are illustrated in Fig. 3.5. Comparing A1 and A2, it is evident that the performance of SVM trained on the synthetically balanced data is significantly higher than that of the baseline classifier, by both metrics, TSS and HSS2. This

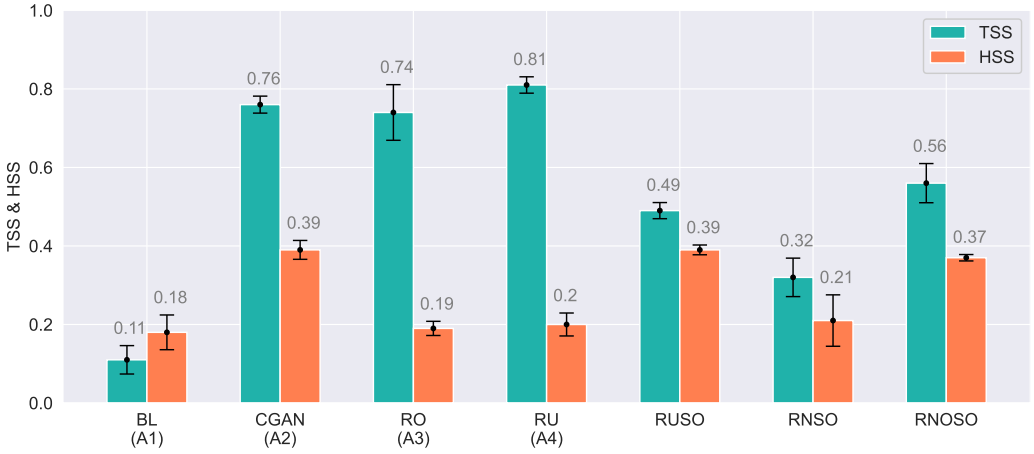

Figure 3.5 The bar plot compares CGAN's synthetically generated data (A2) with the other group A experiments listed in Table 3.1. The choice of the last-value statistic in A makes our results comparable with the naïve random synthetic oversampling methods of RUSO, RNSO, and RNOSO purposed in [2]. The reported TSS and HSS2 values are averaged over three separate evaluation trials on Partitions 2, 3, and 5 of SWAN-SF. Partition 4 is not involved in this experiment. Error bars show the standard deviation of the obtained TSS/HSS2 values.

observation confirms that the model generally performs best when classes in the training dataset are roughly equal in size. Specifically, the CGAN classifier results in a five-fold improvement compared to the baseline experiment in terms of TSS (an increase from 0.11 to 0.76). The HSS2 shows an over one-fold improvement (from 0.18 to 0.39). The HSS2 improvement in A2 compared to A3 and A4 is also significant; from 0.19 to 0.39. TSS, however, remains roughly stagnant in these cases, which is simply due to the difference in what the two metrics measure. It is crucial to note that while balancing the data seems to be the main reason for the significant improvement in performance from A1 to A2, it would not have happened by balancing the dataset with unrealistic flaring instances. This is the main takeaway from our synthetically generated samples that we are evaluating through A2

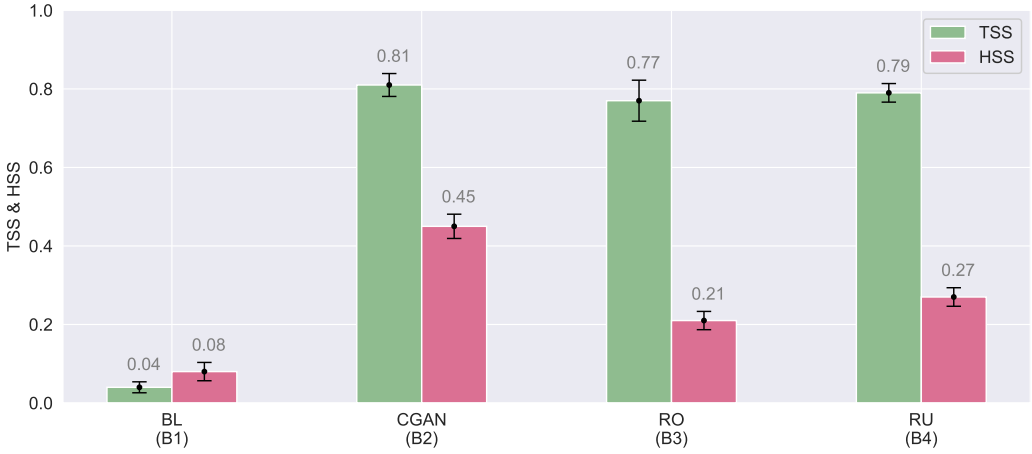

Figure 3.6 The bar plot compares CGAN's synthetically generated data (B2) with the other group A experiments listed in Table 3.1. The reported TSS and HSS2 values are averaged over three separate evaluation trials on Partitions 2, 3, and 5 of SWAN-SF. Partition 4 is not involved in this experiment. Error bars show the standard deviation of the obtained TSS/HSS2 values.

experiment.

Furthermore, compared to the statistic-based oversampling methods purposed in [2], the CGAN-based method achieves a significant improvement in terms of TSS while maintaining HSS2 at its highest value, i.e., 0.39. Overall, the experiment results show that our method can produce a better flare forecasting performance than the random sampling-based methods or the statistic-based oversampling methods.

Next, we examine the forecasting performance of the group B experiments, as shown in Fig. 3.6. In these experiments, we observe that B2 achieves the highest TSS and HSS2. The result shows that the CGAN model can successfully learn the median and standard deviation of real multivariate time series samples.

Putting together the results shown in Figs. 3.5 and 3.6, we demonstrated that our method has multiple advantages compared to other remedies. First, comparing to the random oversampling method (A3 and B3), the CGAN-based method can bring new information through generating realistic synthetic samples instead of duplicating existing samples. Second, comparing to the random undersampling strategy (A4 and B4), the CGAN-based approach can produce unlimited synthetic samples. Thus, more data provides a path towards training more powerful machine learning models. This significantly benefits flare forecasting models based on deep neural networks. Third, comparing to the statistic-based oversampling methods (RUSO, RNSO, and RNOSO), the CGAN-based method can learn the descriptive statistics of the real MVTS samples and, therefore, generate realistic samples. All in all, we can so far conclude that CGAN algorithm can be used to remedy the imbalance issue of MVTS flare datasets. What we have not yet examined, however, is the temporal characteristics of the synthetic time series, and whether they are realistic beyond their median and standard deviation summaries. Next, we put this question to the test.

3.3.5 Examining Synthetic Time Series v.s. Over-/Under-Sampling

In this section, we examine the effectiveness of synthetic samples in time series format. For the experiments in the Group C, we use the same setting of training datasets with the Groups A and B mentioned in Section 3.3.4. The forecasting results of experiments in Group C are reported in Fig. 3.7. We observe that the model in C2 trained on the dataset balanced with the synthetic samples beats the models trained in C1 and C3, in terms of both TSS and HSS2 scores. The experiment C2 shows a 31% improvement in terms of TSS comparing to

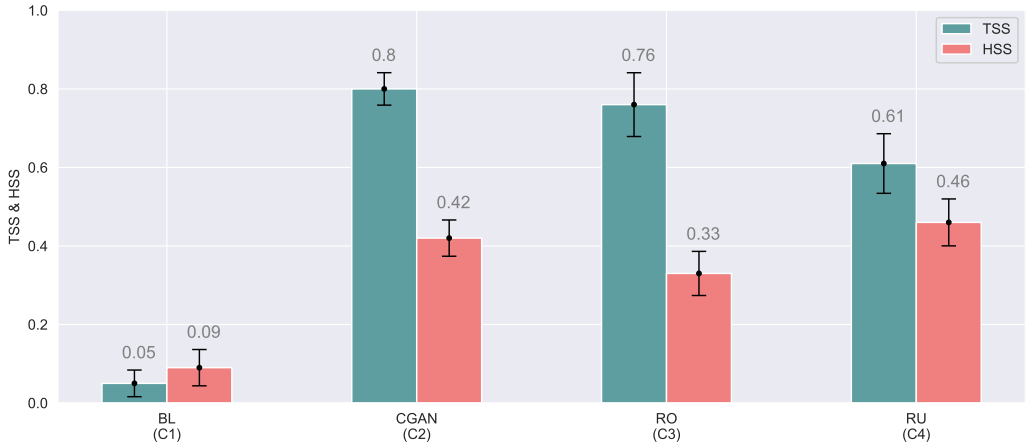

Figure 3.7 The bar plot compares CGAN's synthetically generated data (C2) with the other group C experiments listed in Table 3.1. The reported TSS and HSS2 values are averaged over three separate evaluation trials on Partitions 2, 3, and 5 of SWAN-SF. Partition 4 is reserved for validation of the hyperparameters. Error bars show the standard deviation of the obtained TSS/HSS2 values.

the model trained in C4. Although the model in experiment C2 does not obtain the highest HSS2 score, it still gives a comparable performance. The experimental result validates our assumption that adding informative synthetic samples to balance the training dataset can result in a more robust forecasting model.

3.3.6 Examining Incremental Incorporation of Synthetic Time Series

To further demonstrate the effectiveness of the synthetic multivariate time series, we conduct another experiment to show how varying the number of incorporated synthetic samples affects the forecasting performance. More specifically, we fix the number of real flaring and non-flaring samples in the training dataset, and gradually add synthetic flaring samples while monitoring the model's performance on the test set. As illustrated in Fig. 3.8, we conduct

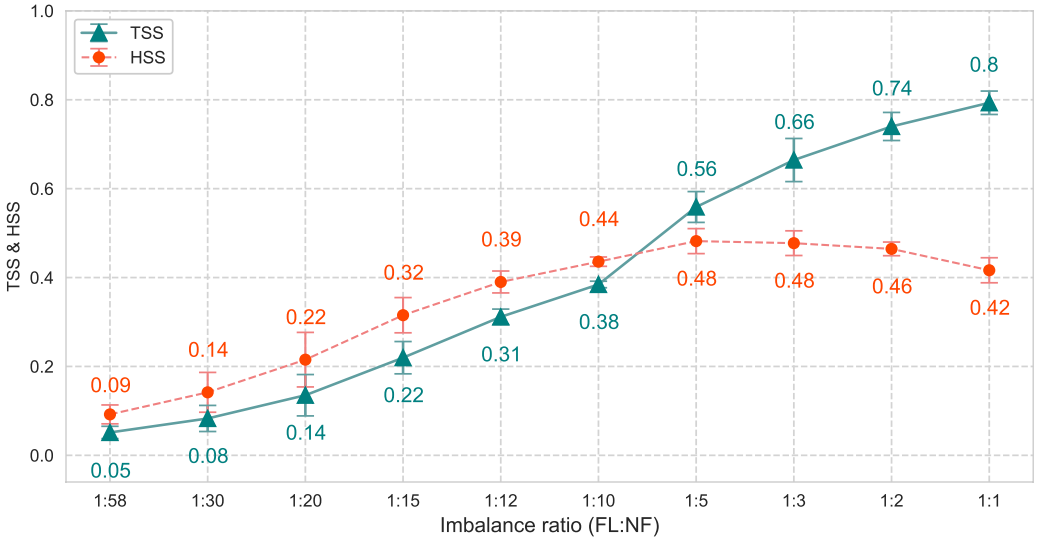

Figure 3.8 The plot illustrates the gradual impact of reducing the imbalance ratio of the training set on performance, by incrementally adding synthetic flaring samples. The reported TSS and HSS2 values are averaged over three separate evaluation trials on Partitions 2, 3, and 5 of SWAN-SF. Partition 4 is reserved for validation of the hyperparameters. Error bars show the standard deviation of the obtained TSS/HSS2 values.

ten experiments by varying the imbalance ratios of the training dataset from 1:58 to 1:1. The ratio of 1:58 is the original imbalance ratio of Partition 1, including $1,254$ real flares and $72,238$ non-flaring samples.

Through observing the result, we can see that the performance generally increases as we reduce the imbalance ratio using our synthetic multivariate time series data. While the strict increase of TSS values indicates that the incorporated synthetic time series are of high quality (when compared with the real time series), we notice that the HSS2 values slightly decline at the very end. Familiar with the different behavior of these two metrics, we believe this is caused due to lack of a per-experiment hyperparameter tuning. In other words, the

added synthetic time series eventually made the default hyperparameters ineffective and consequently the model suboptimal. This change seems to have been overlooked by TSS, but not by HSS2, which is the main reason for using them as a couple. Overall the results show that the trained CGAN model can indeed generate realistic multivariate time series samples.

We would like to recapitulate that our main objective is to show the effectiveness of CGAN as a possible remedy to the class-imbalance issue on SWAN-SF. Therefore, we do not claim the superiority of this approach over any other existing methods, nor do we infer that our findings can be extended to any other multivariate time-series datasets. To this end, we did not include multiple datasets, and we did not compare the performance of CGAN with other GAN-based algorithms. Instead, we kept our focus on evaluating the contribution of CGAN-generated synthetic MVTS of SWAN-SF, and the reliableness of the generated time series for machine learning use.

3.4 Conclusion

In this chapter, we show the usage of the conditional generative adversarial network (CGAN) to perform data-informed augmentation of multivariate time series (MVTS) on a recently released flare forecasting benchmark dataset (SWAN-SF). We tailor several verification methods to show that the generated MVTS samples indeed preserve the distribution of the real physical parameters: (1) we utilize KL-divergence metric to quantify the similarity between the distributions of the real and synthetic data; (2) we use Adversarial Accuracy to moni-

tor the performance of CGAN directly; (3) we use the synthetic MVTS samples to balance our dataset and compare the classification performance with that trained on the original data, and that on the dataset that was balanced by other oversampling, undersampling, and statistic-based synthetic oversampling methods such as RUSO, RNSO, and RNOSO. The results showed that the CGAN-based approach can remarkably boost flare forecasting performance in terms of TSS and HSS2. Therefore, we consider that the CGAN method is an effective remedy for mitigating the class imbalance issue in flare forecasting, and it provides a preliminary attempt to generate meaningful synthetic physical features.

CHAPTER 4

EXAMINING EFFECTS OF CLASS IMBALANCE ON CONDITIONAL GAN TRAINING

In this chapter, we explore the impact of class imbalance on the quality and diversity of synthetic samples generated by conditional generative adversarial networks (CGANs). While CGANs have demonstrated remarkable success in generating realistic image samples using well-processed and balanced benchmark datasets like MNIST and CIFAR-10 [61; 80], real-world applications such as fraud detection, diabetes diagnosis, and solar flare prediction often involve imbalanced data distributions. The well-known class-imbalance issue can also have a profound effect when training CGAN on imbalance datasets, as in [8], the authors stated that traditional GANs cannot be employed to generate minority-class images from an imbalanced dataset.

The aforementioned works mentioned in Section 2.1.2 attempt to address the imbalance issue at the algorithm level, either by employing Autoencoder to learn latent features or by modifying objective functions during the training procedure [8; 9]. Our obejctive is to investigate the issue of class imbalance inherent to GAN training at the data-level, showing that how the imbalance in the training set has a negative effect on the performance of CGANs and the ineffectiveness of common remedies for training GANs on imbalanced datasets, such as oversampling and undersampling.

Furthermore, we introduce a novel approach, the Two-stage CGAN, designed to enhance the quality of minority-class samples when training CGANs on imbalanced datasets. To demonstrate its effectiveness, we conduct experiments using MNIST to generate realistic

images.

4.1 Methodology

We utilize the Conditional Generative Adversarial Network (CGAN), as outlined in Section 3.1, for two primary reasons: Firstly, CGAN enables us to control the category of generated samples, allowing us to address the class imbalance problem by generating samples from minority classes. Secondly, compared to vanilla GANs [73], CGAN offers more stable and efficient training processes. Building on the CGAN framework, we introduce a novel approach named Two-stage CGAN to tackle the inherent challenges of class imbalance during model training.

The proposed pipeline consists of three steps. To begin with, if the original set is unbalanced, we use random undersampling to reduce it to a smaller, more balanced set (i.e., Training-set-1 in Fig. 4.1) and then train the first CGAN model ($CGAN_1$) on this equally represented dataset. After completing $CGAN_1$ training, we can generate synthetic minority class samples, resulting in Synthetic-set-1. The reason for performing undersampling and generating the Synthetic-set-1 dataset based on it is that we discovered that the $CGAN_1$ can create synthetic samples of minority classes with acceptable quality. In the intermediary stage, the original set and Synthetic-set-1 are merged to create Training-set-2 in Fig. 4.1, a balanced and much larger set. This dataset is then used to train the second CGAN model ($CGAN_2$). Again, we generate synthetic minority class samples to construct the Synthetic-set-2. In the final phase, the Original Set and Synthetic-set-2 will be combined to form the

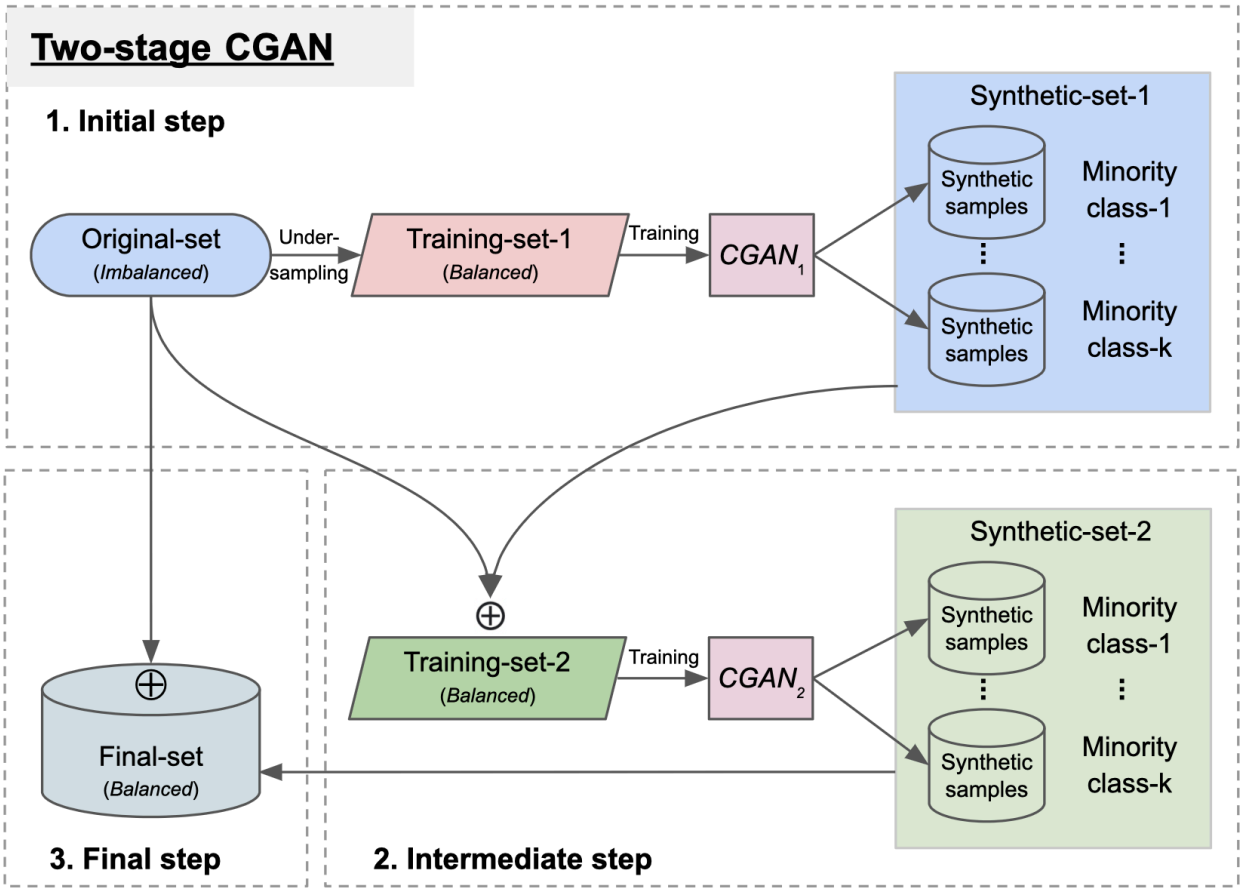

Figure 4.1 The Two-stage CGAN framework consists of three steps: (1) undersampling Original-set and training the $CGAN_1$ model on it to form Synthetic-set-1 for minority classes; (2) merging Original-set and Synthetic-set-1 to training the $CGAN_2$ model to produce Synthetic-set-2 for minority classes; and (3) combining Original-set and Synthetic-set-2 to obtain Final-set for subsequent applications.

final training set (i.e., Augmented-set in Fig. 4.1) for subsequent applications.

4.2 Experiments

4.2.1 Experimental Design

We conducted the experiments on the MNIST dataset, introduced in Section 2.4.1. The original dataset comprises 10 classes, each with approximately 6,000 training images. For brevity, we used a subset of the original MNIST and performed necessary resampling operations to meet experimental requirements. Specifically, we selected five digit classes out of ten, where '0', '1', '2' are considered majority classes and '3', '4' as minority classes, detailed in Table 4.1. In addition, we manually generate five different datasets to evaluate the efficacy of CGAN models trained on them. The dataset-A is derived directly from the original MNIST, which has approximately 6,000 samples per class and is balanced. The dataset-B is created based on A by reducing the minority classes of '3' and '4' to 500 and 100 samples, respectively. We chose 500 and 100 because we wish to examine two distinct imbalance ratios, which are approximately 1:12 and 1:60. If the assumption that the class imbalance issue affects the performance of CGAN models holds true, we consider two common resampling strategies in practice: oversampling and undersampling. The dataset C is created by duplicating and rebalancing existing samples of classes '3' and '4' with majority classes. We can also determine if the overfitting issue resulting from oversampling the underrepresented classes is affecting the sample quality. The dataset D is generated by removing the existing samples of majority classes to align their size with the size of minority classes. The dataset E differs from the dataset C in that it was oversampled using Two-stage CGAN, a newly devised framework. Instead of duplicating existing samples, we rebalance the dataset by

adding 5,500 and 5,900 synthetic samples, respectively, to the minority classes of '3' and '4'.

Table 4.1 The table lists five datasets intended to assess the performance of CGAN training. A is directly taken from the original MNIST. B is produced by reducing the minority classes of '3' and '4' to 500 and 100 samples, respectively, based on A. C and D are obtained by employing oversampling and undersampling strategies to B. E is the dataset that has been augmented on B using Two-stage CGAN.

Dataset	Type	Digit Class					Total
		0	1	2	3	4	
A	Balanced (Baseline)	5923	6742	5958	6131	5842	30596
B	Imbalanced	5923	6742	5958	(6131→) 500	(5842→) 100	19223
C	Oversampling (OS)	5923	6742	5958	(500→) 6000	(100→) 6000	30623
D	Undersampling (US)	(5923→) 100	(6742→) 100	(5958→) 100	(500→) 100	100	500
E	Two-stage CGAN	5923	6742	5958	500_{real}+ $5500_{synthetic}$	100_{real}+ $5900_{synthetic}$	30623

4.2.2 Experimental Settings

We evaluate the performance of CGAN model with the same hyper-parameter configuration across different experiments, setting the latent space dimension to 3, the learning rates to 0.1, the batch size to 32, and the LSTM hidden size of 100. The models were trained with 500 epochs. Empirically, we use the Adam Optimizer for the generator and the Gradient Descent Optimizer for the discriminator. The CGAN model is implemented based on the TensorFlow 2.1 library [76].

4.2.3 Model Selection

For the sake of simplicity, we only display the FID score distribution of the CGAN trained on dataset-A in Fig. 4.2 when performing model selection based on FID scores. We review the checkpoints every 25 epochs, between the 200th and 500th epochs. We conclude that the 300th epoch is a reasonable option given the trade-off between performance and computational cost. We determine the 300th checkpoint in evaluation with experiments A, B, C, and E because the dataset size variation is insignificant. Since the experiment D dataset is much smaller than other datasets, we repeat the FID-based model selection process for it, and we choose to use the 900th checkpoint in the evaluations and analyses that follows.

4.2.4 Examining Two-stage CGAN on Image Generation

Using the setup shown in Table 4.1, we trained CGAN models on each of the five datasets separately. Fig.4.3 shows examples of the output from these models. By looking at the outputs in subplot (A), it is evident that a CGAN trained on a balanced training set is capable of

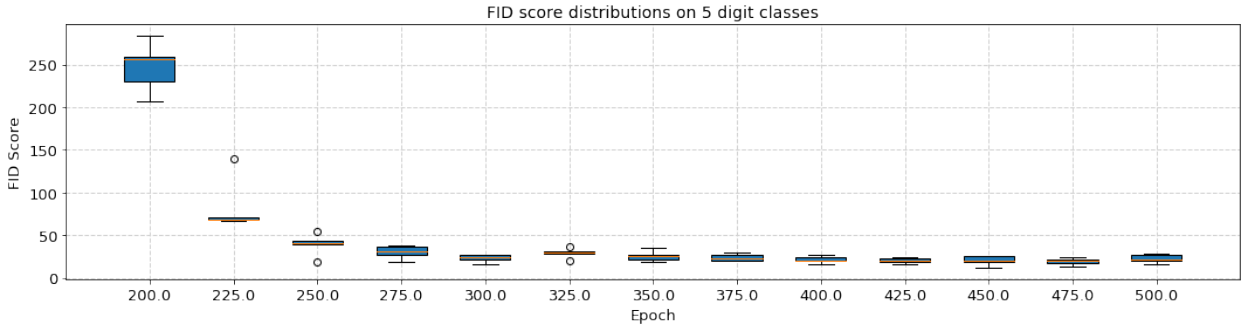

Figure 4.2 The box plots depict the distributions of FID scores for five digit classes (i.e., '0', '1', '2', '3', and '4') as calculated by the CGAN model trained on dataset-A. The x-axis represents the models per 25 epochs between the 200th and 500th epochs, and the y-axis represents the corresponding FID scores for each class. This metric is considered the selection criterion for models.

producing acceptable synthetic samples for all classes. However, in the subplot (B), we discovered that the generated samples of class '4' are of lesser quality, whereas we can generate samples of comparable quality for other classes using A. This confirms the assumption that the imbalance ratio in the training set can affect the performance of a GAN, i.e. that GANs give more attention to the majority classes in practice. In scenario (C), we observe that both '3' and '4' synthetic samples have low diversity and low quality. This may be because the random oversampling strategy typically involves duplicating samples exactly to expand the data space, which may lead to the overfitting issue. Therefore, balancing the training set by randomly oversampling minority class samples cannot enhance the performance of the CGAN and generate high-quality synthetic samples. In (D), we can see that the diversity of minority classes is better than the results in (C), although this strategy can put the loss of important concepts at risk. The lower-quality outputs are caused by insufficient training data. The subplot (E) depicts the synthetic samples generated by Two-stage CGAN, which

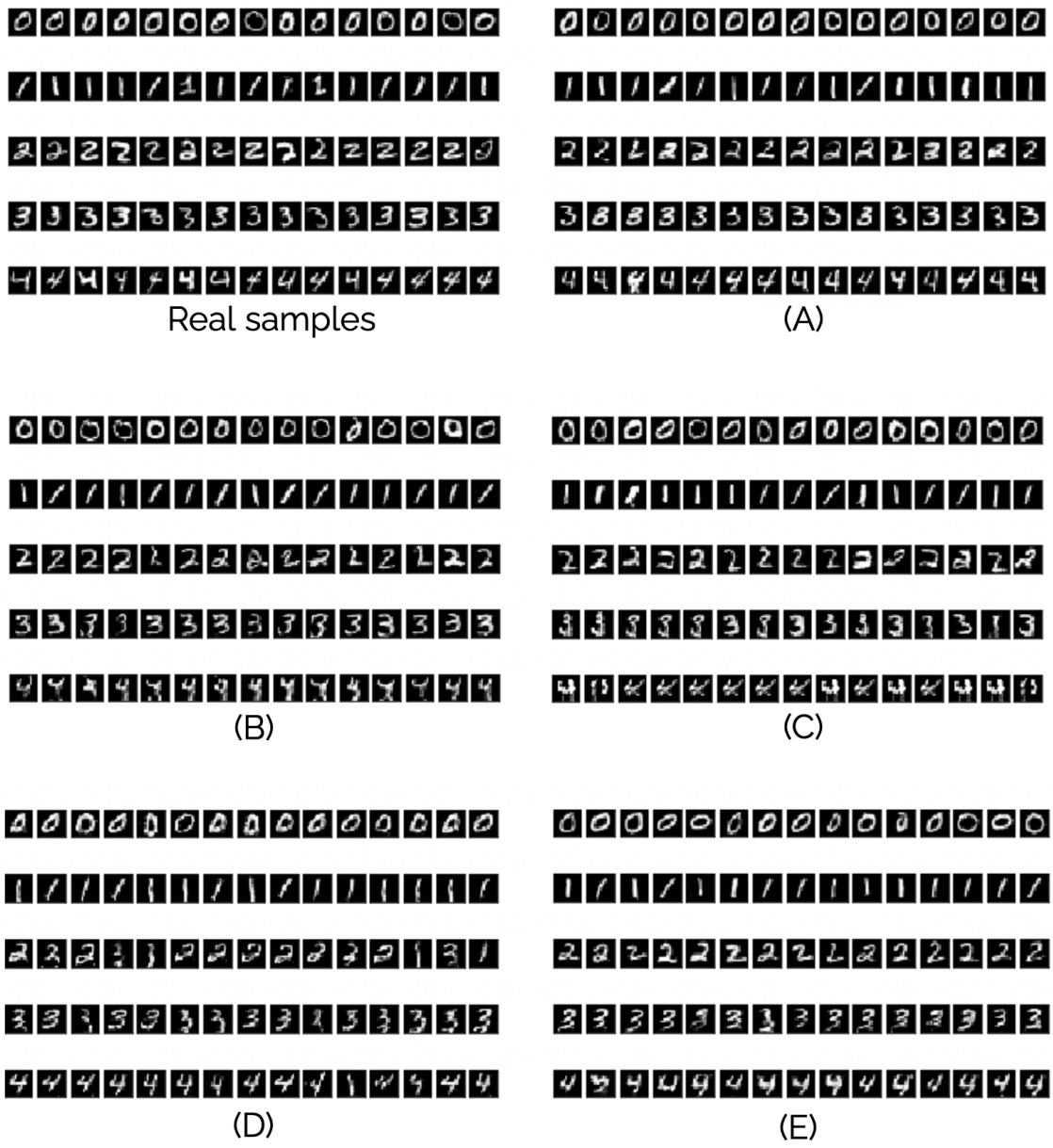

Figure 4.3 The diagram shows real samples and synthetic samples generated by CGAN models trained on datasets in Table 4.1.

enhance both quality and diversity simultaneously.

The FID score is then utilized to quantitatively assess the similarity between the real and synthetic samples. Specifically, 1,000 real samples per class are selected at random, and

1,000 synthetic samples per class are generated using CGAN models trained on five training sets. The results of the FID are shown in Table 4.2. Row-A displays the FID scores of five classes for a balanced training set, which can be interpreted as the baseline similarity between real and synthetic samples. Row B is the result of an imbalanced training set. We discovered that the FID scores for the majority classes (i.e., '0', '1', and '2') are lower than A while the scores for the minority classes (i.e., '3' and '4') are higher than A, with means of 32.89 and 86.79, respectively. The '4' digit class, which has the greatest imbalance in our design (approximately 1:60), is especially affected. There are two possible explanations for why the digit class of '3' is not significantly affected: (1) Because the class of '3' is not the rarest, the weights in the generator for generating '3's receive more training opportunities than the weights for generating '4's; (2) because the digits '2' and '3' are naturally more similar, feeding sufficient samples of '2' into the training process can aid the training process of '3'. The FID scores of the oversampling strategy as a remedy for class imbalance are displayed in Row C. The increased FID scores of both minority classes (i.e., '3' and '4') indicate less similarity between real and synthetic samples. This is more evident for '4', indicating that oversampling cannot mitigate the data deficiency issue and result in a well-trained synthetic data generator for minority classes. Row D displays the FID score of using the undersampling strategy as the class imbalance remedy, which results in higher FID scores than Row A. However, given that the training set is balanced, the variances in FID are not that great. In D, the FID score of the digit '4' is 45.81, which is lower than in B and C, indicating that the synthetic samples of '4' are more similar to real samples of '4'. Therefore,

we are considering utilizing this advantage to generate synthetic samples of minority classes to supplement the imbalance dataset (i.e., dataset B), and then training a final CGAN model on a larger and more balanced training set, yielding the result of Row E. Observing the FID results for Row E, we can see that it achieves the lowest FID scores of all classes, indicating the proposed framework provides a significant improvement over typical oversampling and undersampling techniques utilized for class imbalance remediation.

Table 4.2 The table provides a summary of the FID evaluation outcomes from five experiments. Each FID score is determined by comparing 1,000 actual and 1,000 synthetic samples of the same class. Five separate simulations are performed to calculate the final results, guaranteeing the correctness of the assessment.

$FID_{(mean \pm std)}$	Digit - 0	Digit - 1	Digit - 2	Digit - 3	Digit - 4
A	21.44 ± 0.32	15.81 ± 0.20	27.47 ± 0.36	26.62 ± 0.45	24.99 ± 0.55
B	15.28 ± 0.38	11.07 ± 0.26	15.11 ± 0.15	32.89 ± 0.23	86.79 ± 0.50
C	15.12 ± 0.19	14.16 ± 0.36	25.46 ± 0.44	51.75 ± 0.81	142.59 ± 0.59
D	55.36 ± 0.48	30.31 ± 0.28	59.81 ± 0.39	53.73 ± 0.57	45.32 ± 0.46
E	12.44 ± 0.28	10.03 ± 0.39	17.11 ± 0.52	10.31 ± 0.23	9.81 ± 0.19

4.3 Conclusion

In this chapter, we show how the imbalance in the training set has a negative effect on the performance of GANs. In addition, we show the ineffectiveness of common remedies for training GANs on imbalanced datasets, such as oversampling and undersampling. To address these challenges, we propose a novel solution called Two-stage CGAN aimed at enhancing the quality of samples from minority classes in image contexts. Our experimental results confirm that this framework significantly improves the quality of synthetic samples. However, if try to apply the proposed framework to the context of time series generation, we face the challenge of lacking a widely accepted metric for evaluating synthetic time series.

However, applying the proposed framework to time series generation presents a challenge due to the lack of a widely accepted metric for evaluating synthetic time series. In the next chapter, we introduce a novel metric named the Fréchet Fourier-transform Auto-encoder Distance (FFAD), which leverages Fourier transform and Auto-encoder techniques to assess the quality and diversity of generated time series samples. By using FFAD as the model selection criterion, we further demonstrate the effectiveness of Two-stage CGAN in time series generation, leading to practical applicability and performance improvements in solar flare forecasting tasks.

CHAPTER 5

FFAD: A NOVEL METRIC FOR ASSESSING TIME SERIES-BASED GENERATIVE MODELS

The success of deep learning-based generative models in generating realistic images, videos, and audio has raised an important question: how can we effectively evaluate the quality of synthetic samples? While the Fréchet Inception Distance (FID) serves as the standard metric for assessing generative models in image synthesis, a comparable metric for time series data is conspicuously absent. This gap in assessment capabilities arises from the lack of widely accepted feature vector extractors pretrained on benchmark time series datasets. To address the challenges in evaluating time series quality, particularly using Fréchet Distance, this study proposes a novel solution: the Fréchet Fourier-transform Auto-encoder Distance (FFAD). By leveraging the Fourier transform and Auto-encoder, FFAD offers a promising approach to distinguishing samples from different classes and evaluating the quality and diversity of generated time series samples. Our experimental results demonstrate the potential of FFAD, contributing to the ongoing efforts to enhance assessment methodologies in deep learning-based generative models. Finally, we extend the Two-stage CGAN framework to synthetic time series generation, and conduct experiments in solar flare forecasting to validate the efficacy.

5.1 Background

5.1.1 Fourier Transform

The Fourier transform, an integral mathematical technique applied extensively in signal processing, mathematics, and diverse scientific disciplines, is employed to analyze and depict functions or signals within the frequency domain. Its primary objective is the dissection of complex signals into constituent sinusoidal components. This process involves the decomposition of time-domain signals into their corresponding frequency components, providing a comprehensive understanding of the varied frequency contributions comprising the signal, as shown in Fig. 5.2.

Numerous Fourier Transform implementations have been proposed. The Discrete Fourier Transform (DFT) is widely employed in digital signal processing when dealing with discrete, sampled data. The Fast Fourier Transform (FFT) has an improved computation time compared to the straightforward evaluation of the DFT, making it widely used in applications such as signal processing, image analysis, and many scientific computations [81]. In this project, we prioritize the use of FFT due to its superior computational efficiency, enabling rapid analysis and processing of complex datasets.

5.1.2 Auto-encoder

Auto-encoders, a class of neural networks, have gained significant attention in the domain of deep learning and unsupervised learning. A typical auto-encoder consists of an encoder

[1]Image source: www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft

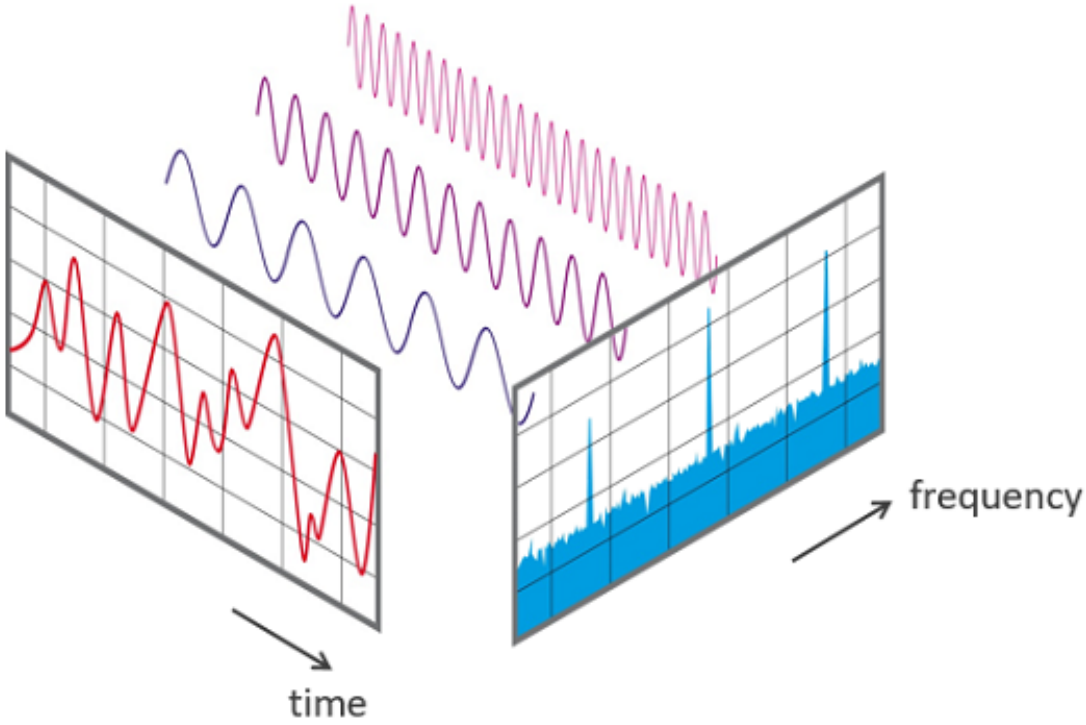

Figure 5.1 An example of viewing a time signal in both the time and frequency domains utilizing Fourier Transform. [1]

and a decoder. The encoder compresses input data into a lower-dimensional representation, while the decoder reconstructs the original input from this code. During training, the network learns to minimize the difference between the input and the reconstructed output, facilitating the extraction of meaningful features in the encoded representation. To address specific data types, Convolutional Auto-encoders are customized for image data through the integration of convolutional layers [82]. Similarly, Recurrent Auto-encoders are devised for sequential data, such as time series, employing recurrent units [83]. In this work, we utilize the Recurrent Auto-encoder, considering frequency components as sequential data.

[2]Image source: https://www.compthree.com/blog/autoencoder/

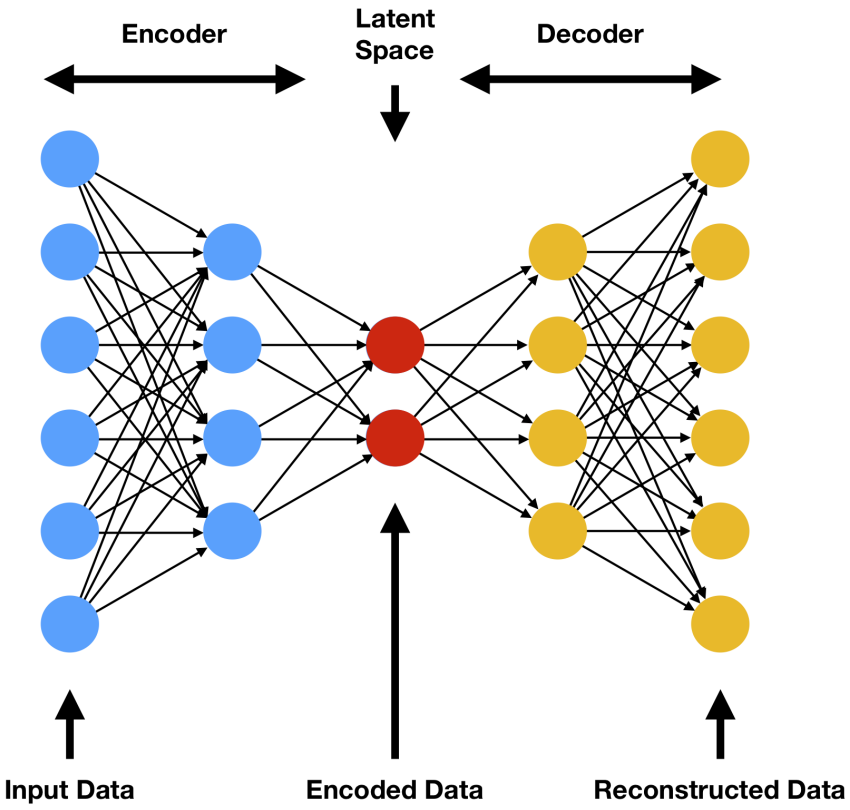

Figure 5.2 The macro-architecture of an Auto-encoder consists of an "encoder" followed by a "decoder." The encoder maps the input data to a low-dimensional "latent" space, while the decoder attempts to reconstruct the low-dimensional representation back to the original high-dimensional space. [2]

5.2 Methodology

Our main goal is to train an Auto-encoder utilizing a variety of time series datasets. Following this, we aim to utilize the Encoder component to generate a lower-dimensional representation for any time series, whether from real-world datasets or synthetic ones produced by generative models like conditional GANs. Furthermore, we will assess the dissimilarity between distributions using the FFAD score, which compares pairs of time series datasets, whether they are from different categories or involve comparisons between real and synthetic

samples.

To effectively train an Auto-encoder for time series data, the foremost challenge to address is handling datasets with varying sequence lengths. Within the UCR dataset collection, the time series datasets can range in length from 15 to 2,844 data points, with an average length of 537 time steps, as depicted in Fig. 5.3. To tackle this significant variability, we utilize Fourier Transformation as a preprocessing step for the original UCR time series. This choice is guided by two primary reasons: (1) Ensuring all time series data with a consistent input length for training RNN-based auto-encoder. Maintaining a uniform input length eliminates the need for padding the variable-length inputs, resulting in increased time efficiency. (2) Shifting from time domain to frequency domain with Fourier Transform while preserving essential features. This transformation enables us to represent the original time series by selecting a suitable number of sine components, which are not only appropriate but also fewer in number compared to the original sequence length.

Consider a collection of time series datasets denoted as $D = \{d_1, d_2, ..., d_n\}$, as shown in (A) of Fig. 5.4. Each individual dataset d_i has its own length len_i and number of samples $|d_i|$. As a result, each d_i can be represented as a matrix with dimensions $[|d_i|, len_i]$. Through the utilization of the Fourier transformation, we can convert each dataset from the time domain to the frequency domain. Assuming a selection of m frequency components, each datasets yields a Fourier-transform (FT) representation matrix (Z_{d_i}) with the shape of $[|d_i|, m]$. Choosing the same m value for all datasets enables the concatenation of the FT representation matrices into a larger matrix (Z_D) with the shape of $[N, m]$ (i.e., $N = \sum_{i=1}^{n} |d_i|$),

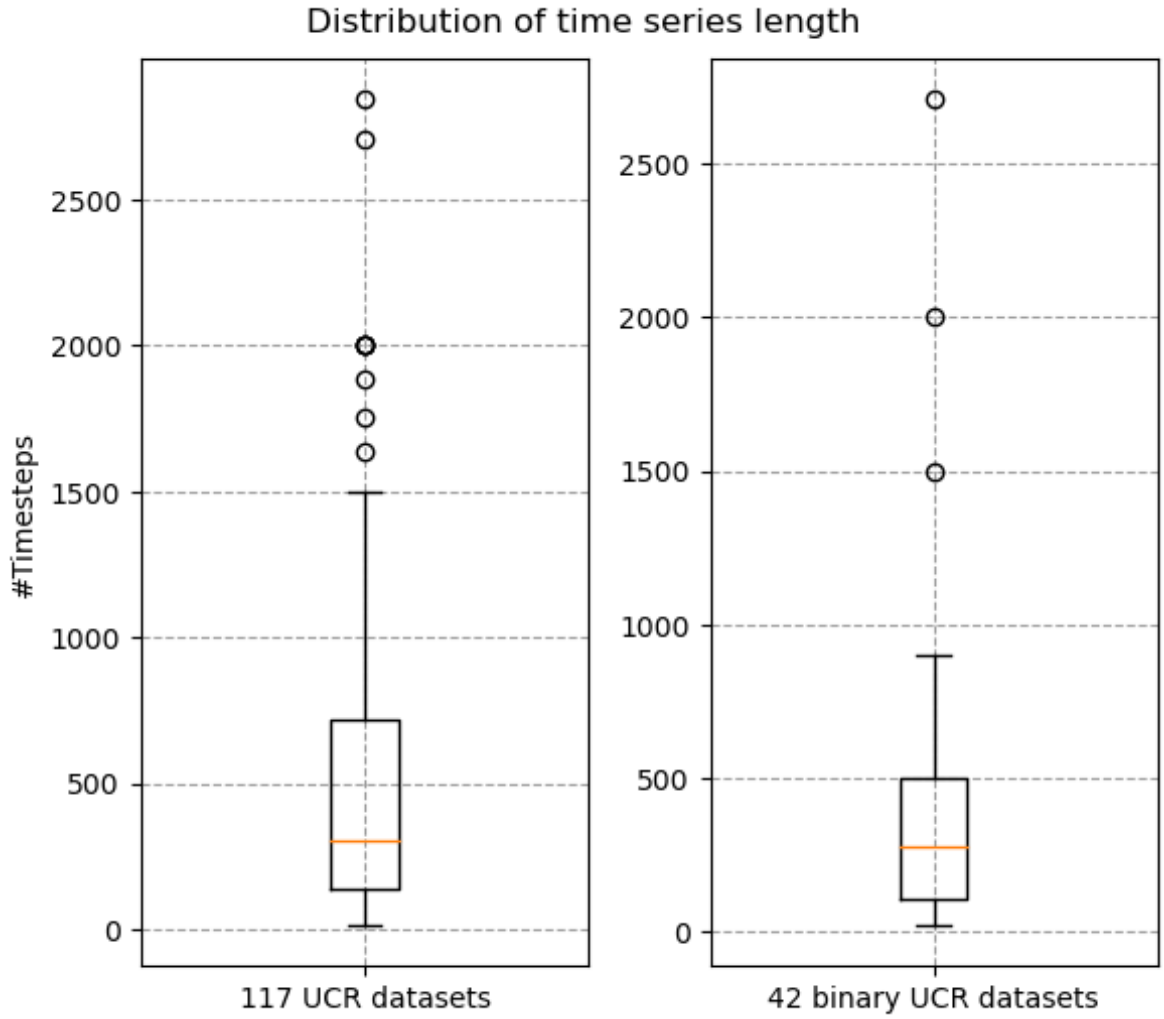

Figure 5.3 The plot illustrates the distribution of time series lengths in the UCR dataset collection. These time series vary in length from 15 to 2,844 data points, with an average length of 537 time steps.

as shown in Eq. 5.1. Within Z_D, each element (e.g., $z_{i,j}^k$) represents k as the dataset index, i as the sample index within each dataset, $|di|$ indicating the total number of samples for a specific dataset, and j as the index of frequency components.

$$Z_D = [Z_{d_1}, Z_{d_2}, ..., Z_{d_n}] = \begin{bmatrix} z^1_{1,1} & z^1_{1,2} & ... & z^1_{1,m} \\ z^1_{2,1} & z^1_{1,2} & ... & z^1_{1,m} \\ ... & ... & ... & ... \\ z^1_{|d_1|,1} & z^1_{|d_1|,2} & ... & z^1_{|d_1|,m} \\ z^2_{1,1} & z^2_{1,2} & ... & z^2_{1,m} \\ ... & ... & ... & ... \\ z^n_{1,1} & z^N_{1,2} & ... & z^N_{1,m} \\ ... & ... & ... & ... \\ z^n_{|d_n|,1} & z^n_{|d_n|,2} & ... & z^n_{|d_n|,m} \end{bmatrix} \xrightarrow{shape} [N, m] \tag{5.1}$$

An aspect that requires investigation concerns the characteristics of Fourier transform results. These results manifest as complex numbers, encompassing crucial information about both magnitude and phase. Since we employ Keras, which primarily supports real-valued computations for neural networks, and doesn't have native support for complex numbers in its core operations, so the pre-preocessing for complex numbers is needed. Considering each row in Z_D as list of frequency components $z = [z_1, z_2, ..., z_m]$ (where $z_i = a + bj$), we separate the real (a) and imaginary parts (bj) by organizing them into a two-dimension array $z^{'} = [[a_1, b_1], [a_2, b_2], ..., [a_m, b_m]]$, characterized by a shape of $[m, 2]$. The ultimate shape of the matrix Z_D will be $[N, m, 2]$.

In the implementation of the Recurrent Auto-encoder, we employ the Gated Recurrent Unit (GRU) for both the Encoder and Decoder components. GRU is selected for its enhanced performance in processing long sequences, by minimizing the risk of the gradient vanishing problem [84]. We adopt a mini-batch approach to train the Auto-encoder effectively. During each iteration, a batch comprising a batch-size of samples (referred to as X) from the matrix Z_D serves as input for the Encoder component. The output of the Encoder, denoted as $Y = \text{Encoder}(X)$, represents a significantly compressed representation compared

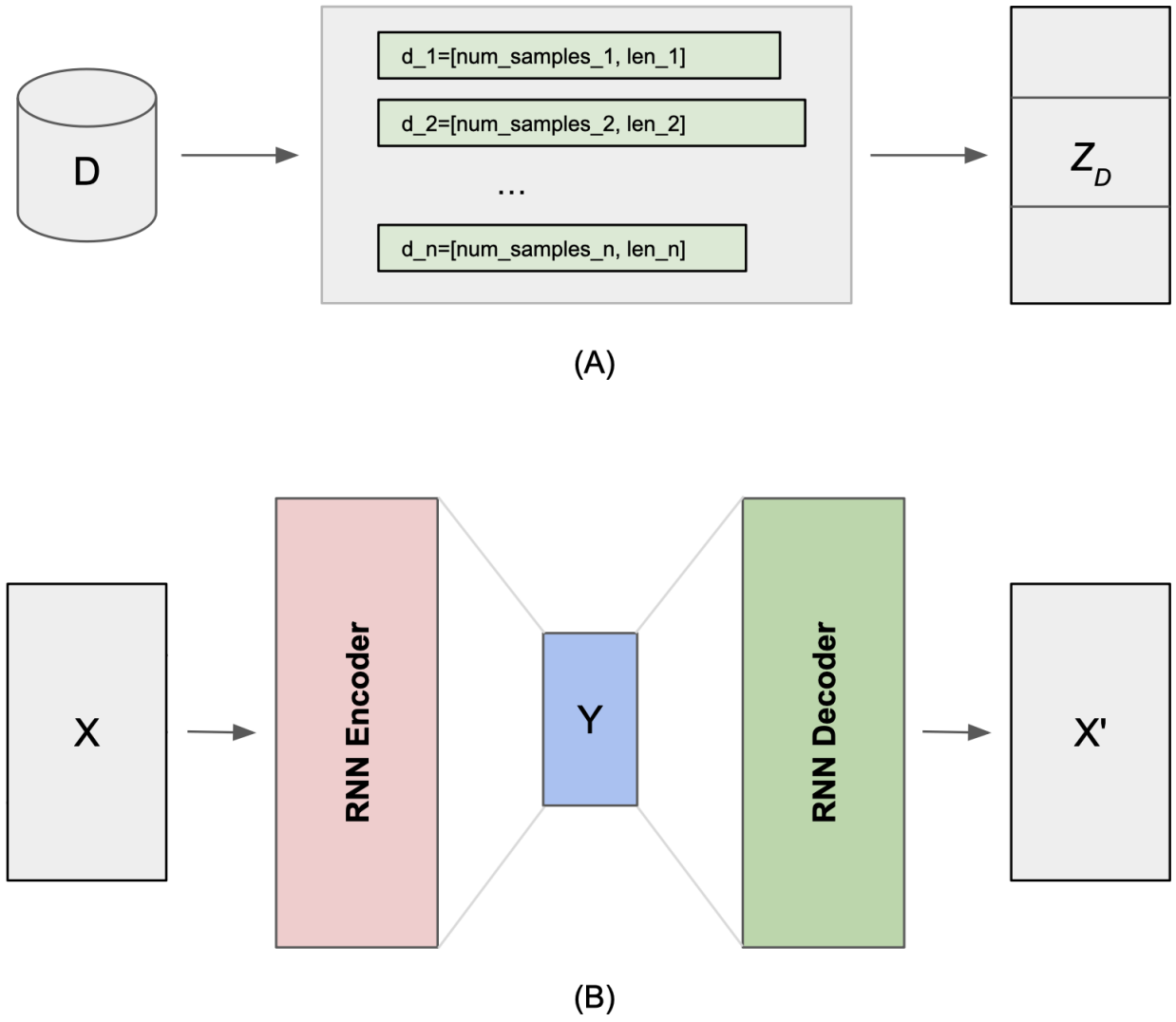

Figure 5.4 Sub-figure (A) illustrates the procedure of employing Fourier Transformation as a preprocessing step for the original time series data, ensuring a consistent length for all datasets. Sub-figure (B) outlines the training procedure of the autoencoder.

to X. For the Encoder training, we require another integral component known as the Decoder. This Decoder takes Encoder(X) as its input and generates an output represented by $X' = \text{Decoder}(Y) = \text{Decoder}(\text{Encoder}(X))$. The primary goal of the Decoder is to reconstruct the initial input data X. Therefore, the overarching objective function of the entire

model is to minimize the error between X' and X. In practice, we utilize the Mean Square Error (MSE) as the training criterion, as shown in Eq. 5.2.

$$\begin{aligned} \text{Loss} = \left(\frac{1}{|batch|} \sum_{i=1}^{|batch|} (X'_i - X_i)^2 \right) \rightarrow \text{Minimized}, \\ where \quad X'_i = \text{Decoder}(Y_i) = \text{Decoder}\left(\text{Encoder}(X_i)\right) \end{aligned} \tag{5.2}$$

After completing the training of the auto-encoder, the focus shifts to retaining solely the Encoder component. Taking a binary dataset d_i as a case study, we have two sample sets: S_{pos} representing the positive class and S_{neg} signifying the negative class. Employing the Encoder, which has been effectively trained, we generate encoded representations denoted as Y_{pos} and Y_{neg} for the positive and negative sets, respectively. Then, the FFAD score can be calculated to measure the similarity between S_{pos} and S_{neg}, as illustrated in Eq. 5.3,

$$\text{FFAD Score} = ||\mu_{Y_{\text{pos}}} - \mu_{Y_{\text{neg}}}||^2 + \text{Tr}(\Sigma_{Y_{\text{pos}}} + \Sigma_{Y_{\text{neg}}} - 2\sqrt{\Sigma_{Y_{\text{pos}}}\Sigma_{Y_{\text{neg}}}}) \tag{5.3}$$

Here, Y_{pos} and Y_{neg} serve as the encoded representations, while $\mu_{Y_{\text{pos}}}$ and $\mu_{Y_{\text{neg}}}$ correspond to the vector magnitudes of Y_{pos} and Y_{neg}, respectively. The function $Tr(.)$ denotes the trace of the matrix, with $\Sigma_{Y_{\text{pos}}}$ and $\Sigma_{Y_{\text{neg}}}$ representing the covariance matrices of Y_{pos} and Y_{neg}. Lower values within this equation indicate a higher similarity between the two input sets.

Furthermore, we implement the Inverse Fourier Transform to verify the reconstruction capability of the transformed data. As shown in Algorithm 1, the Inverse Fourier Transform function requires two parameters: $frequencies$ and $length$. The $frequencies$ parameter is

Algorithm 1 Inverse Fourier Transform

Input: frequencies // a list of freq components

length // the original time series' length

Output: rec_ts // The reconstructed time series.

1: Set rec_ts to an array of zeros with length elements

2: Set index to an array containing values of [0, length) with increments of 1.0 $i = 1$ to *length*

3: Set rec_ts to 0

4: Set index[i] to i * (2 * pi) / length

k, p in enumerate($frequencies$)

5: If k is not equal to 0, then multiply p by 2

6: Separate the real and imaginary components of p as a+bj $j = 1$ to *length*

7: Add a * cos(k * index[j]) to rec_ts[j]

8: Subtract b * sin(k * index[j]) from rec_ts[j]

anticipated to be a list or array containing the Fourier transform coefficients of the sequence. On the other hand, *length* signifies the length of the original time series. The goal of the Inverse Fourier Transform is to merge these Fourier coefficients to reconstruct the original time series.

5.3 Experiments

We conducted experiments on two widely recognized public datasets: UCR and SWAN-SF (as mentioned in Section 2.4), with the aim of addressing the following objectives:

- (1) Determining the optimal number of frequency components using Fourier Transform to convert time series data from the time domain to the frequency domain, thereby standardizing their representation and mitigating the effects of varying sequence lengths.
- (2) Demonstrating the effectiveness of the compressed representations obtained by training a general Auto-Encoder on a diverse set of time series datasets (UCR and SWAN-SF).
- (3) Assessing the consistency of the FFAD score with two statistical model selection methods introduced in Section 3.2.
- (4) Using the FFAD score as an evaluation metric to distinguish between data from different classes and to assess the realism of generated samples across multiple classes.
- (5) To extend the Two-stage CGAN to synthetic time series generation and validate its efficacy through experiments aimed at improving solar flare forecasting performance.

5.3.1 Transforming Data with Fourier Transform

We conducted an experiment to evaluate the capacity of various frequency components in representing the original sequences through Fourier transformation. Our investigation involved assessing the reconstruction capability across different numbers of frequency components:

$\{1, 2, 3, 5, 10, 15, 20, 30\}$. To conduct this analysis, we selected the SWAN-SF dataset, focusing on four parameters from partition-1. This selection was deliberate for two key reasons: (1) SWAN-SF represents a real-world dataset ideal for reconstruction studies, and (2) its extensive collection of time series encapsulates the inherent complexity often observed in such data.

To assess the reconstruction capability of the transformed data, we employed the Inverse Fourier Transform (Inverse-FT) as described in Section 5.2. Fig. 5.5 illustrates the reconstruction examples for the ABSNJZH parameter using varying numbers of frequencies, ranging from 1 to 30. When the number of frequency components is between 1 and 5, the Inverse-FT can only approximate the major trend of the input time series in a coarse manner. However, with the utilization of more frequency components (i.e., 10, 15, 20, 30), the restored time series becomes more refined, and the Inverse-FT can fully recover the input time series using all 30 components (considering the input time series comprises 60 time steps).

To conduct a comprehensive evaluation of the reconstruction performance, we employ Mean Square Error (MSE) as the evaluation metric, as discussed in Section 5.2. The corresponding results are depicted in Fig. 5.6. It is evident from the figure that the Mean Squared Error (MSE) decreases as more frequency components are incorporated into the reconstruction process. We conclude that employing 20 components achieves a favorable equilibrium between compressed representation and reconstruction capability for subsequent experiments.

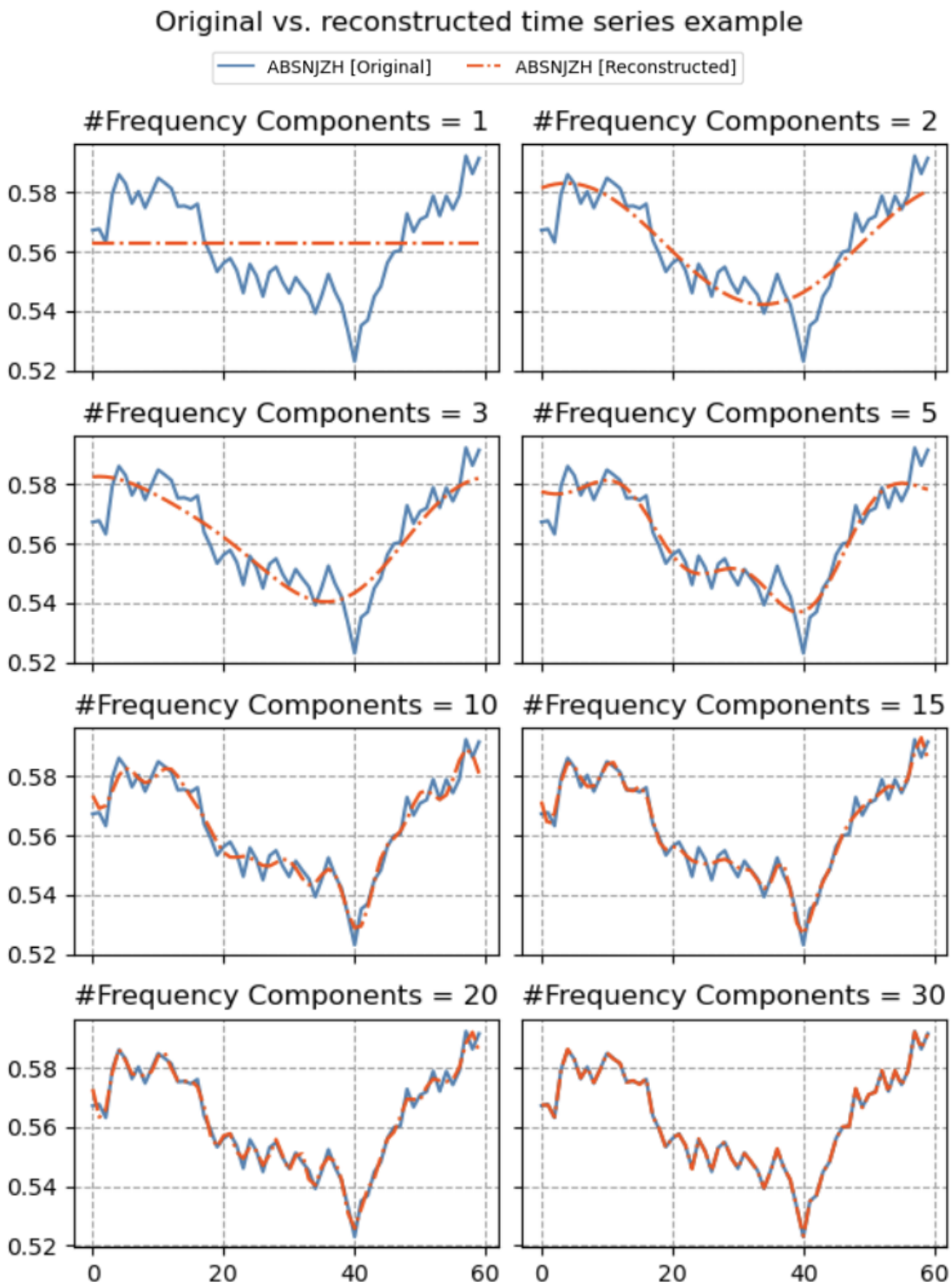

Figure 5.5 Shows the original time series and reconstructed time series utilizing different number of frequency components. This example is sourced from Partition 1 of SWAN-SF.

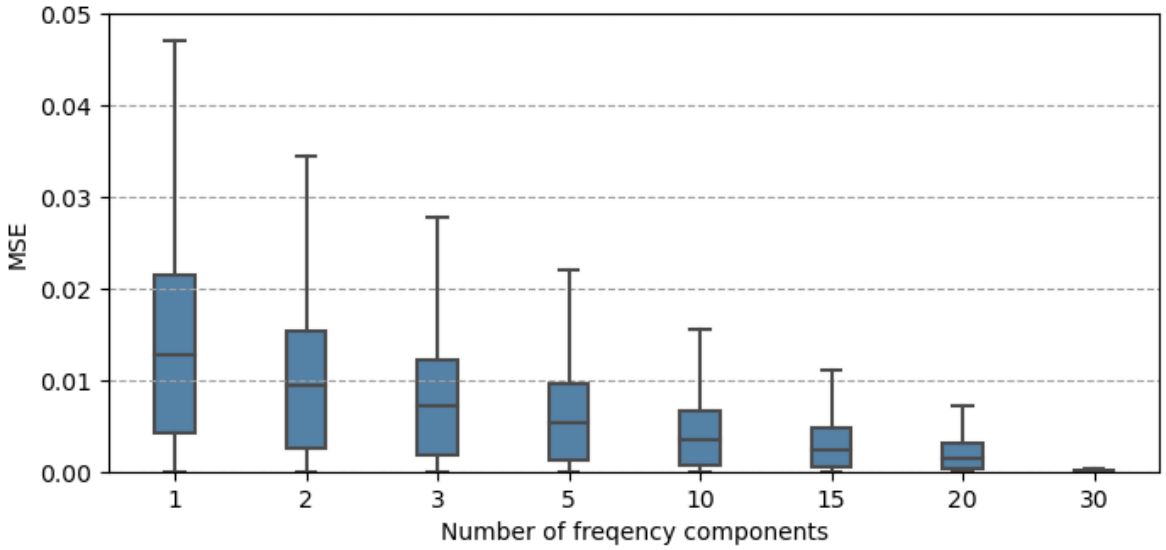

Figure 5.6 The results provide a comprehensive evaluation of the reconstruction performance on Partition 1 of SWAN-SF .

5.3.2 Training Auto-encoder and Model Selection Criteria

To train a unified Auto-encoder, we combined 97 UCR datasets with the SWAN-SF dataset. The training dataset comprises the training sets from the 97 UCR datasets, as well as partition-1 of SWAN-SF, which includes 4 parameters. Similarly, the testing dataset includes the testing sets from the 97 UCR datasets, along with partition-2 of SWAN-SF, containing 4 parameters. We tune the performance of Auto-encoder model by setting different hyper-parameters, i.e. GRU hidden sizes: $\{5, 10, 20, 30\}$, learning rates: $\{0.1, 0.01, 0.001, 0.0001\}$, batch sizes: $\{256, 512, 1, 024\}$. Empirically, we concluded our optimal hyper-parameter setting with the GRU hidden size of 20, the learning rate of 0.001, the batch size of 512. The model was trained with $5,000$ epochs.

To perform model selection during the Auto-encoder's training, we utilize Mean Square Error (MSE) as the evaluation metric over every 500 epochs. More specifically, we randomly

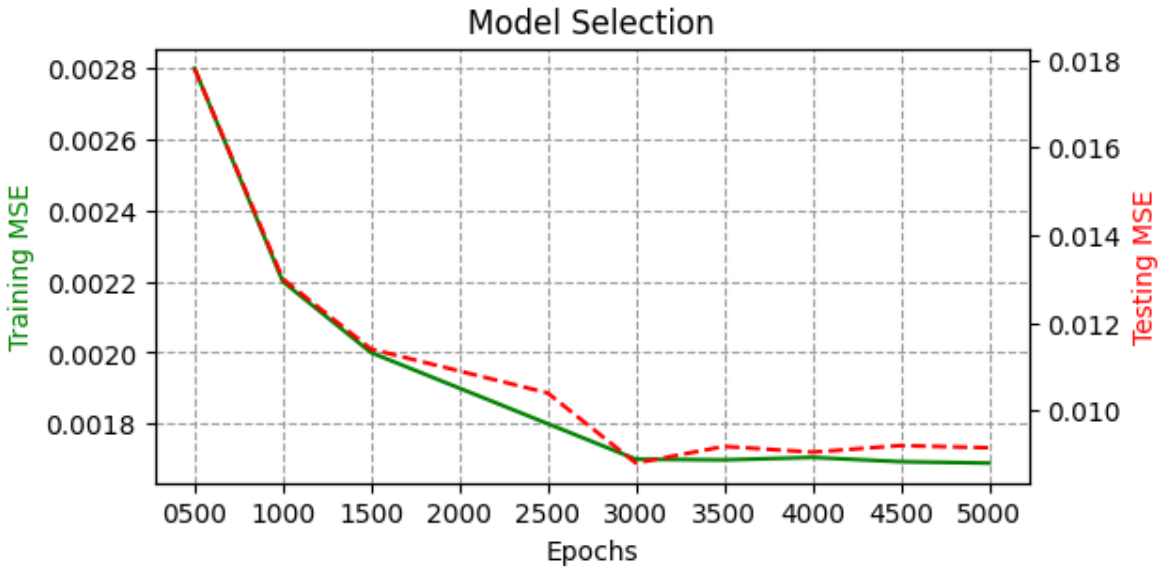

Figure 5.7 Shows the procedure of selecting the Auto-encoder model by calculating Mean Square Error (MSE) as the evaluation metric every 500 epochs, and identifies that the optimal model is achieved at the 3,000th epoch.

select 10,000 from both the training and testing sets to compute the MSE, and the outcomes are depicted in Fig. 5.7. Analyzing the training and testing curves, we identify that the optimal model is achieved at the 3,000th epoch. Additionally, Fig. 5.8 shows six pairs of original and reconstructed time series examples, selected from the testing test of UCR and SWAN-SF. These examples illustrate how well the Auto-encoder preserves the essential characteristics of the original data, showcasing its ability to effectively reconstruct time series samples.

5.3.3 Using FFAD to Differentiate Same-Class vs. different-Class

In this section, we evaluate the effectiveness of FFAD in distinguishing between samples from the same or different classes. We consider a hypothetical binary dataset divided into

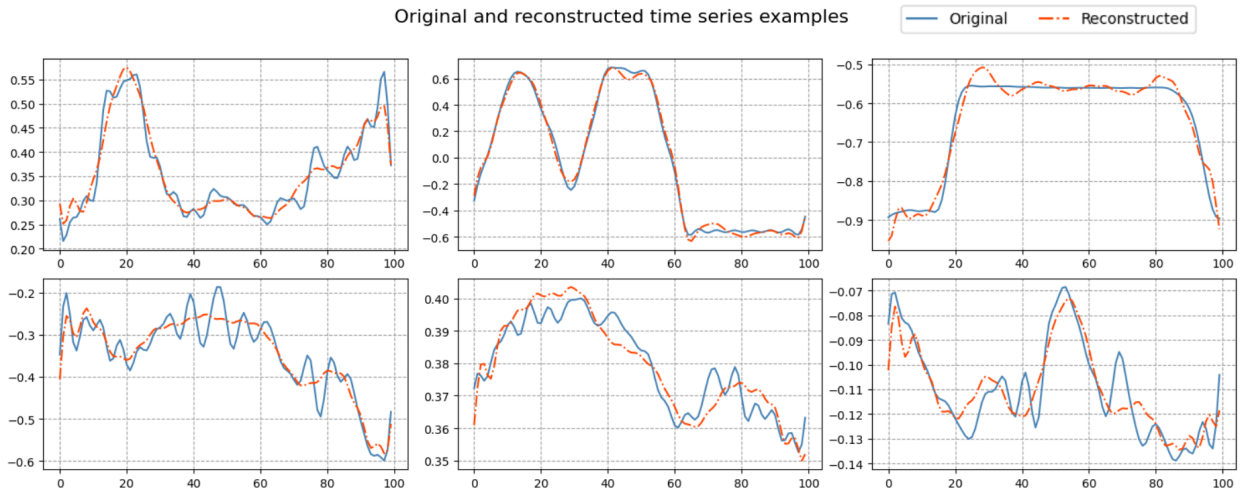

Figure 5.8 Displays six pairs of original and reconstructed time series. The examples are selected from UCR and SWAN-SF datasets.

training and testing sets, each containing two classes (e.g., class-0 and class-1), resulting in four sub-datasets. The FFAD score is computed across these sub-datasets, yielding six scores by pairing any two of them. Notably, two scores stem from comparisons within the same classes (e.g., train-0 vs. test-0 and train-1 vs. test-1), while the remaining four scores arise from comparisons across different classes (e.g., train-0 vs. train-1, train-0 vs. test-1, train-1 vs. test-0, and test-0 vs. test-1). Table 5.1 summarizes the FFAD results for ten binary UCR datasets. Upon observing the results, it is evident that FFAD scores for same-class comparisons are notably lower than those for different-class comparisons. This observation implies the effectiveness of FFAD in distinguishing between samples from the same or different classes.

Table 5.1 The table reported the FFAD scores for ten binary UCR datasets, with each score representing the average from five repeated experiments. The actual scores are presented in scientific notation, multiplied by 10^{-5}.

Binary	Same-class		Different-class			
Datasets	train-0 vs. test-0	train-1 vs. test-1	train-0 vs. train-1	train-0 vs. test-1	train-1 vs. test-0	test-0 vs. test-1
1. BeetleFly	67.81	64.07	174.01	146.74	335.66	243.5
2. BirdChicken	40.34	18.11	43.9	52.76	64.43	71.39
3. ECG200	202.86	28.7	1431.86	1336.32	1298.21	1090.2
4. ECGFiveDays	0.75	5.09	24.45	15.08	20.84	11.1
5. GunPoint	22.75	15.66	426.84	314.97	327.61	219.7
6. Lightning2	1.44	1.1	4.47	3.71	4.47	1.86
7. Strawberry	1.37	10.74	81.88	48.43	102.79	62.11
8. TwoLeadECG	10.93	4.52	47.0	29.38	34.54	18.59
9. Wafer	2.52	1.17	11.21	11.16	14.77	11.32
10. Yoga	3.3	4.15	21.26	13.13	15.89	12.8

5.3.4 Using FFAD to Differentiate Real vs. Synthetic Samples

In this section, we extensively evaluated FFAD's ability to distinguish between real and synthetic samples using the SWAN-SF dataset. Our main goal was to generate high-quality synthetic samples for this real-world dataset. Specifically, we utilized 1,254 real flare samples and 1,254 real non-flare samples of the physical parameter TOTUSJH from Partition 1 of SWAN-SF. Subsequently, we generated an equal number of synthetic flare and non-flare samples using the CGAN model.

Table 5.2 presents the FFAD score comparisons on the SWAN-SF dataset for both same-class and different-class scenarios. Here, 'r' and 's' denote real and synthetic data, respectively. Our analysis yielded several key findings: Firstly, FFAD scores for same-class scenarios (i.e., FL_r *vs.* FL_s and NF_r *vs.* NF_s) were notably lower compared to those for different-class scenarios, indicating higher similarity or realism of synthetic samples generated by the CGAN model relative to real samples. Secondly, all different-class scenarios exhibited similar FFAD scores, with FL_r *vs.* NF_r serving as the baseline, suggesting consistency and realism of synthetic samples across different classes. These results indicate the effectiveness of FFAD in assessing the realism of generated samples and verifying their class information, establishing it as a crucial metric for evaluating the fidelity of generated time series data in our study.

Table 5.2 The table reported FFAD scores comparison on the SWAN-SF dataset for both same-class and different-class scenarios. Here, FL and NF denote flare and non-flare samples, respectively, with 'r' and 's' indicating real and synthetic data.

SWAN-SF	Same-class		Different-class			
Dataset	FL_r *vs.* FL_s	NF_r *vs.* NF_s	FL_r *vs.* NF_s	FL_s *vs.* NF_s	FL_s *vs.* NF_r	FL_r *vs.* NF_r
FFAD scores	0.008	0.003	0.482	0.440	0.458	0.499

5.3.5 Consistency of Various Model Selection Strategies

In the last section, we would like to compare FFAD with two model selection methods that have previously introduced in Section 3.2: one using distributions of statistical features and another using Adversarial Accuracy. It is valuable to examine the consistency among these various model selection methods. This examination also serves to validate the effectiveness of FFAD in assessing the realism of generated samples.

We conducted experiments using 1,254 real flare time series samples from Partition 1 of SWAN-SF and 1,254 synthetic samples generated by the CGAN model. Due to FFAD's current limitation to one-dimensional inputs, our CGAN training focused solely on the physical parameter TOTUSJH. The model was trained for 300 epochs with checkpoints saved every 5 epochs, resulting in a total of 60 checkpoints, with other settings same with Section 3.3.1. For each checkpoint, we calculated the KL-divergence, Adversarial Accuracy, and FFAD score. To streamline computation, we utilized the mean as the representative statistical feature

for calculating KL-divergence and Adversarial Accuracy during each checkpoint evaluation. As illustrated in Fig. 5.9, we observed that all three scores achieved their lowest values between epochs 201 and 250, which is in line with our previous findings. This consistency further validates that the FFAD score aligns well with the other two evaluation methods we employed.

Furthermore, we can compare three model selection methods. Specifically, both the KL-divergence and Adversarial Accuracy methods utilize statistical features such as mean, median, and standard deviation as inputs. The KL-divergence method aims to measure the similarity between the distributions of real and synthetic samples, but it can yield misleadingly low scores if the model overfits or duplicates samples. Addressing this, the Adversarial Accuracy aims for a score of approximately 0.5, indicating that real and synthetic samples are indistinguishable. In contrast, FFAD utilizes time series inputs, providing a more comprehensive assessment based on global information. It leverages a pre-trained model to extract features from time series, providing more informative features than simple statistics (e.g., mean and standard deviation). While Adversarial Accuracy is computationally intensive, requiring distance computation between each sample and others, FFAD and KL-divergence scores are efficient as they focus on distribution perspectives. In addition, FFAD can be utilized independently for assessing synthetic samples, whereas KL-divergence often requires Adversarial Accuracy as a complement. Overall, FFAD not only shows consistency with statistic-based methods but also balances comprehensive evaluation and computational efficiency, making it a promising metric for model selection and synthetic data assessment

in time series analysis.

5.3.6 Exploring Optimal Synthetic Data Generation Strategy for Flare Forecasting

In this section, we explore optimal strategies for generating synthetic multivariate time series for flare forecasting. We compare the performance of CGAN models trained using different methodologies on imbalanced time series datasets, focusing on four physical parameters from Partition 1 of the SWAN-SF dataset (as mentioned in Section 2.4.3). This partition includes 1,254 flare samples and 70,984 non-flare samples. To create a balanced training set, we explore three approaches:

- CGAN(OS): Oversampling the 1,254 flare samples to match the 70,984 non-flare samples, resulting in a balanced training set of 144,476 samples. This strategy is detailed in Section 3.3.

- CGAN(US): Undersampling the 70,984 non-flare samples to match the 1,254 flare samples, resulting in a smaller balanced training set of 2,508 samples.

- Two-stage CGAN: Implementing a two-stage CGAN approach (described in Section 4.1) on time series data to create a larger balanced training set. The first stage follows the CGAN(US) strategy, while the second stage combines synthetic samples generated by the first stage CGAN ($CGAN_1$) with the original imbalanced dataset to form a larger balanced training set. This set is then utilized to train another CGAN ($CGAN_2$), which generates high-quality synthetic samples. These synthetic samples are combined with

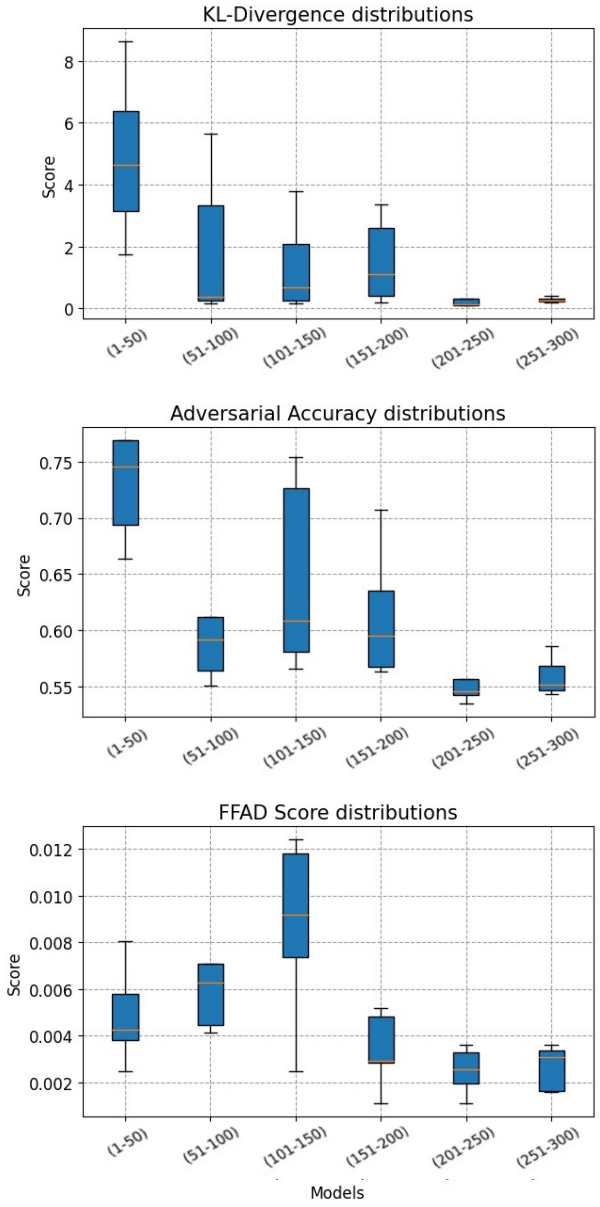

Figure 5.9 The plots shows the distributions of KL-divergence, Adversarial Accuracy and FFAD scores calculated between synthetic and real samples. The CGAN model was trained for 300 epochs, with checkpoints saved every 5 epochs resulting in a total of 60 checkpoints, divided into six groups for creating the boxplots.

the original dataset to form a final training set that is large, balanced, and of high fidelity, suitable for subsequent applications.

We evaluate the performance of these approaches through a flare forecasting task. For CGAN(OS) and CGAN(US), we create corresponding datasets and train CGAN models using the settings described in Section 3.3.1. The main difference lies in the number of training epochs: CGAN(OS) is trained for 300 epochs, while CGAN(US) requires approximately 15,000 epochs, adjusted for training set size differences. In the Two-stage CGAN experiment, we conduct 15,000 epochs in stage-1 and 300 epochs in stage-2. Throughout all experiments, we employ the FFAD score as the model selection criterion.

Using the synthetic time series of the flare class generated from each scenario, we rebalance Partition 1 of the SWAN-SF dataset. We then train an SVM based on each rebalanced training set using the settings mentioned in Section 3.3.1. Finally, forecasting is performed on Partitions 2, 3, and 5, and results for TSS and HSS2 are reported.

In Fig.5.10, the Two-stage CGAN outperforms our previous CGAN(OS) strategy, achieving a higher TSS score (0.83) while maintaining a similar HSS2 score (0.44). CGAN(US) shows the lowest performance in both metrics, suggesting that undersampling may compromise model effectiveness due to information loss. This finding aligns with the analysis from the image experiment detailed in Section 4.2. Moreover, the comparison between CGAN(US) and Two-stage CGAN underscores the value of the stage-2 data augmentation in enhancing the quality of generated data. Overall, our experiment suggests that the Two-stage CGAN represents a promising strategy for generating high-quality synthetic time series samples

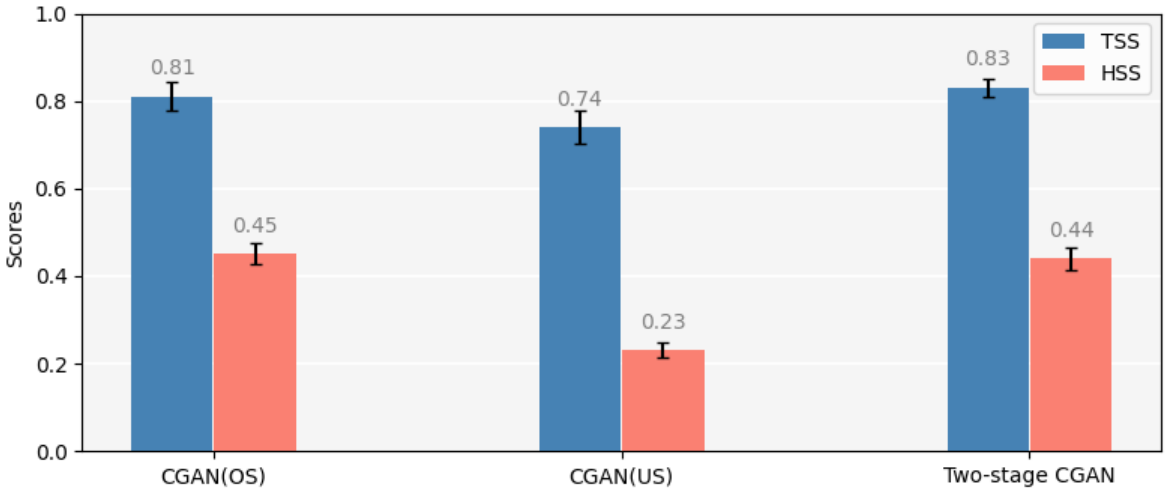

Figure 5.10 The bar plot presents the scores for three different CGAN training strategies evaluated using TSS and HSS2. The three strategies are CGAN(OS) (oversampling), CGAN(US) (undersampling), and Two-stage CGAN. The reported TSS and HSS2 values are averaged over four separate evaluation trials on Partitions 2, 3, and 5 of SWAN-SF. Error bars show the standard deviation of the obtained TSS/HSS2 values.

and may serve as an effective remedy for mitigating class imbalance issue in classification problems.

5.3.7 A Case Study of How Synthetic Data Generation Benefits Flare Forecasting

In this section, we demonstrate how synthetic time series data generation mitigates class imbalance and improves solar flare forecasting performance. We train two SVM classifiers, M1 and M2, using different data rebalancing strategies based on four physical parameters from the SWAN-SF dataset (as described in Section 2.4.3). Classifier M1 is trained on Partition 1, rebalanced using synthetic flare samples generated by Two-stage CGAN. Classifier M2 is trained on Partition 1, rebalanced via random oversampling of flare samples. Both strategies result in a training set comprising 70,984 flare samples and 70,984 non-flare samples.

After training, we test both classifiers on Partition 2 and present the classification results for flare samples (X & M classes) in Fig. 5.11. The scatter plot displays the median values of TOTUSJH on the x-axis and the standard deviation values of TOTUSJH on the y-axis. The data points are colored as follows: grey points represent samples correctly classified by both M1 and M2; red points indicate samples correctly classified only by M1; purple points show samples correctly classified only by M2; and black points denote samples misclassified by both models. The distribution of red points, predominantly found between the grey and black points, indicates that these samples are uniquely classified by M1. This suggests that M1 achieves a better classification boundary generalized to the testing data. Our results indicate that the CGAN approach can effective generate synthetic samples, thereby improving classification performance.

5.4 Conclusion

In this chapter, we introduce a novel metric called the Fréchet Fourier-transform Auto-encoder Distance (FFAD), which seamlessly integrates the Fourier transform and Auto-encoder. We demonstrate that FFAD aligns with statistical feature-based methods for CGAN model selection. Our experimental results further illustrate FFAD's effectiveness in distinguishing between samples from the same or different classes. Next, we employ generative models, such as conditional GANs, in time series generation tasks. Our findings show that FFAD can not only assess the realism of generated samples but also verify if these samples accurately correspond to their class information, thereby establishing itself

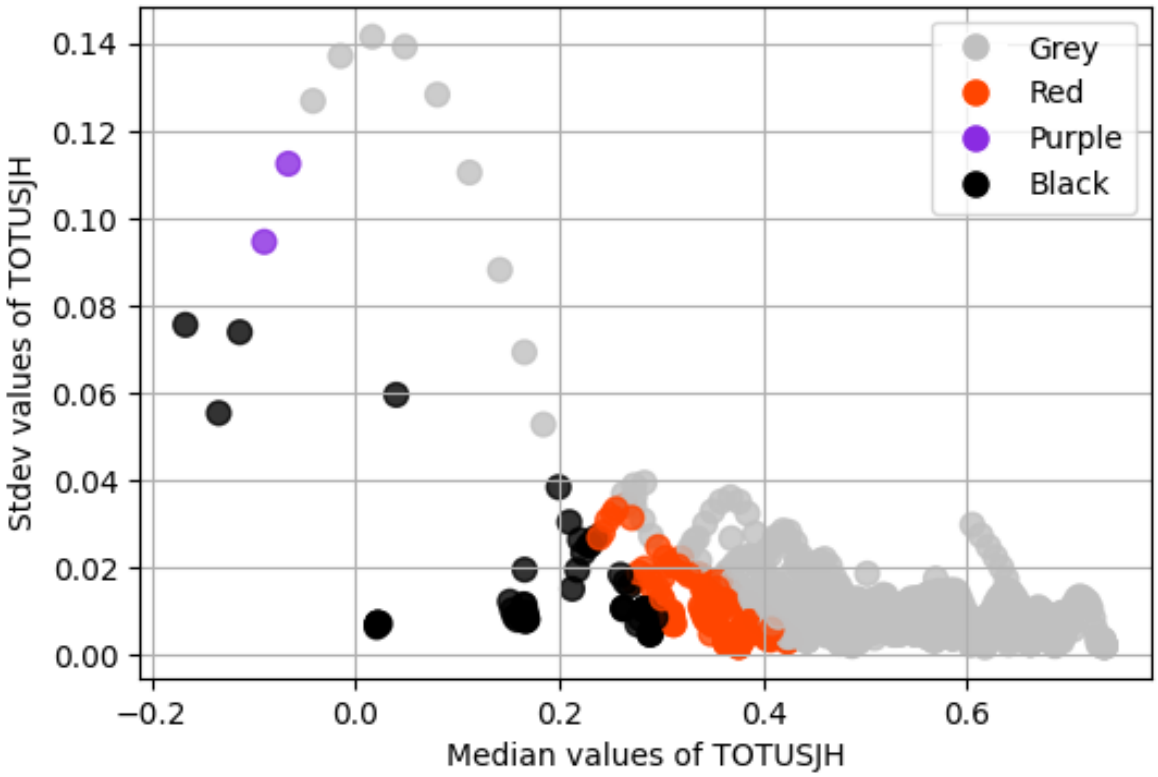

Figure 5.11 The scatter plot illustrates the classification results for flare samples (X and M classes), with the median values of TOTUSJH on the x-axis and the standard deviation values of TOTUSJH on the y-axis.Grey points: Classified correctly by both M1 and M2. Red points: Classified correctly only by M1. Purple points: Classified correctly only by M2. Black points: Misclassified by both.

as a robust metric for evaluating generated time series data. Furthermore, our experiments show that the Two-stage CGAN is the most effective strategy for generating synthetic data and highlight how this synthetic data enhances classification performance in flare forecasting tasks.

CHAPTER 6
CONCLUSION

This dissertation presents my research on using conditional generative adversarial networks (CGANs) to create synthetic multivariate time series (MVTS) samples for flare forecasting. The study employs the SWAN-SF benchmark dataset and develops multiple verification methods to ensure the generated samples accurately reflect real physical parameter distributions. Our experiments demonstrate that the CGAN approach effectively addresses class imbalance issues in flare forecasting. Additionally, we introduce a novel Two-stage CGAN framework to improve CGAN performance on imbalanced datasets. Moreover, we propose FFAD, a new metric for evaluating synthetic time series data. Our results show that FFAD successfully assesses the realism of generated time series samples, suggesting its potential utility in refining existing methodologies for time series generation.

Our work contributes to the broader field of data synthesis and flare forecasting, offering new techniques and evaluation methods that can be applied beyond solar flare prediction to other domains involving imbalanced time series data.

6.1 Future Work

The CGAN-based method provides an promising attempt to generate reliable synthetic dataset, leading to an effective remedy for mitigating the class-imbalance issue. In future, we plan to extend our research from following aspects: First, we are working on a novel CGAN utilizing a Fourier Frequency Component-based generator and a time series-based discriminator. This approach allows us to generate time series data from the frequency domain

perspective, which offers several advantages. By leveraging Fourier frequency components, we can capture and model the inherent periodicities and spectral characteristics of time series data more effectively. Additionally, using a time series-based discriminator ensures that the temporal dependencies and sequential nature of the data are preserved and properly evaluated during training. This dual approach of combining frequency domain insights with time series discrimination provides a robust framework for addressing the complexities of time series data, potentially leading to an improved generative model. Next, we plan to incorporate multi-modal learning with Conditional GAN (CGAN). By integrating various data modalities such as time series data of solar activity and corresponding images from solar observatories, a multi-modal CGAN can capture complex relationships between different data types. The generator network in such a framework learns to generate coherent representations of solar flare events that align with both temporal patterns and visual observations, enhancing the realism and predictive accuracy of generated samples. This approach will facilitates the generation of synthetic data that mirrors real-world solar phenomena [85]. Third is to investigate class imbalance issue in more severe scenarios, such as forecasting coronal mass ejections (CMEs) or solar energetic particle (SEP) events. These phenomena are significantly less frequent than solar flares but can have far more severe impacts on space weather, satellite operations, and terrestrial power grids. The rarity of these events intensifies the class imbalance problem, significantly challenging the development of accurate and reliable forecasting models. Addressing this imbalance is crucial for enhancing the reliability and robustness of predictive models in these high-impact scenarios. Moreover, we hope

to collaborate more with astrophysical specialists, seeking a deeper understanding of the significance of synthetic samples and how they might be used to investigate astrophysical processes. We firmly believe that our cooperation will stimulate new lines of scientific inquiry and result in advances in the understanding of the solar.

REFERENCES

[1] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[2] Maxwell Hostetter and Rafal A. Angryk. First steps toward synthetic sample generation for machine learning based flare forecasting. In Xintao Wu, Chris Jermaine, Li Xiong, Xiaohua Hu, Olivera Kotevska, Siyuan Lu, Weija Xu, Srinivas Aluru, Chengxiang Zhai, Eyhab Al-Masri, Zhiyuan Chen, and Jeff Saltz, editors, *IEEE International Conference on Big Data, Big Data 2020, Atlanta, GA, USA, December 10-13, 2020*, pages 4208–4217. IEEE, 2020.

[3] National Research Council. *Severe Space Weather Events–Understanding Societal and Economic Impacts: A Workshop Report*. The National Academies Press, Washington, DC, 2008.

[4] D. H. Boteler. Geomagnetic hazards to conducting networks. *Natural Hazards*, 28(2):537–561, 2003.

[5] Azim Ahmadzadeh, Berkay Aydin, Manolis K. Georgoulis, Dustin J. Kempton, Sushant S. Mahajan, and Rafal A. Angryk. How to train your flare prediction model: Revisiting robust sampling of rare events. *The Astrophysical Journal Supplement Series*, 254(2):23, May 2021.

[6] A. Ahmadzadeh, B. Aydin, D. J. Kempton, M. Hostetter, R. A. Angryk, M. K. Geor-

goulis, and S. S. Mahajan. Rare-event time series prediction: A case study of solar flare forecasting. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1814–1820, 2019.

[7] A. Ahmadzadeh, M. Hostetter, B. Aydin, M. K. Georgoulis, D. J. Kempton, S. S. Mahajan, and R. Angryk. Challenges with extreme class-imbalance and temporal coherence: A study on solar flare data. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1423–1431, 2019.

[8] Gaofeng Huang and Amir Hossein Jafari. Enhanced balancing gan: Minority-class image generation. *Neural Computing and Applications*, pages 1–10, 2021.

[9] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi. Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*, 2018.

[10] Yang Chen, Dustin J. Kempton., and Rafal A. Angryk. FFAD: A novel metric for assessing generated time series data utilizing fourier transform and auto-encoder. *arXiv preprint arXiv:2403.06576*, 2024.

[11] Russell W Walter. Methods to address extreme class imbalance in machine learning based network intrusion detection systems. 2016.

[12] Tawfiq Hasanin, Taghi M Khoshgoftaar, Joffrey L Leevy, and Richard A Bauder. Severely imbalanced big data challenges: investigating data sampling approaches. *Journal of Big Data*, 6(1):1–25, 2019.

[13] Sara Fotouhi, Shahrokh Asadi, and Michael W. Kattan. A comprehensive data level

analysis for cancer diagnosis on imbalanced data. *Journal of Biomedical Informatics*, 90:103089, 2019.

[14] Georgios Douzas and Fernando Bacao. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with Applications*, 91:464–471, 2018.

[15] Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14:554–562, 2014.

[16] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern recognition*, 40(12):3358–3378, 2007.

[17] Peter D Turney. Types of cost in inductive concept learning. *arXiv preprint cs/0212034*, 2002.

[18] Jason Brownlee. *Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning.* Machine Learning Mastery, 2020.

[19] Ibraheem M Alkhawaldeh, Ibrahem Albalkhi, and Abdulqadir Jeprel Naswhan. Challenges and limitations of synthetic minority oversampling techniques in machine learning. *World Journal of Methodology*, 13(5):373, 2023.

[20] N. Chawla, K. Bowyer, L. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.

[21] Azim Ahmadzadeh, Berkay Aydin, Manolis K. Georgoulis, Dustin J. Kempton,

Sushant S. Mahajan, and Rafal A. Angryk. How to train your flare prediction model: Revisiting robust sampling of rare events. *The Astrophysical Journal Supplement Series*, 254(2):23, may 2021.

[22] Rafal A. Angryk, Petrus C. Martens, Berkay Aydin, Dustin Kempton, Sushant S. Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Xumin Cai, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Michael A. Schuh, and Manolis K. Georgoulis. Multivariate time series dataset for space weather data analytics. *Scientific Data*, 7(1), July 2020.

[23] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1–20, 2018.

[24] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*, 2020.

[25] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei Efros. Everybody dance now. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2019.

[26] Olof Mogren. C-rnn-gan: A continuous recurrent neural network with adversarial training. In *Constructive Machine Learning Workshop (CML) at NIPS 2016*, page 1, 2016.

[27] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*

- *Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[28] Yang Chen, Dustin J. Kempton, Azim Ahmadzadeh, and Rafal A. Angryk. Towards synthetic multivariate time series generation for flare forecasting, 2021.

[29] Ken Chen, Bao-Liang Lu, and James T Kwok. Efficient classification of multi-label and imbalanced data using min-max modular classifiers. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1770–1775. IEEE, 2006.

[30] Muhammad Atif Tahir, Josef Kittler, and Fei Yan. Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognition*, 45(10):3738–3750, 2012.

[31] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[32] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 214–223. JMLR.org, 2017.

[33] A. Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2016.

[34] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2180–2188, Red Hook, NY, USA, 2016.

Curran Associates Inc.

[35] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1558–1566. JMLR.org, 2016.

[36] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[37] Cristóbal Esteban, Stephanie L. Hyland, and G. Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *ArXiv*, abs/1706.02633, 2017.

[38] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 464–483, New York, NY, USA, 2020. Association for Computing Machinery.

[39] Chi Zhang, Sanmukh R. Kuppannagari, Rajgopal Kannan, and Viktor K. Prasanna. Generative adversarial network for synthetic time series data generation in smart grids. In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6, 2018.

[40] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5508–5518, 2019.

[41] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks.

ArXiv, abs/1708.05509, 2017.

[42] Rui Huang, Shu Zhang, Tianyu Li, and R. He. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2458–2467, 2017.

[43] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 405–415, Red Hook, NY, USA, 2017. Curran Associates Inc.

[44] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[45] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

[46] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5908–5916, 2017.

[47] C. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised map inference for image super-resolution. In *International Conference on Learning Representations*, 2017.

[48] C. Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, J. Totz, Zehan Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017.

[49] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8183–8192, 2018.

[50] Shota Haradal, Hideaki Hayashi, and Seiichi Uchida. Biosignal data augmentation based on generative adversarial networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 368–371, 2018.

[51] L. Simonetto. Generating spiking time series with generative adversarial networks : an application on banking transactions. 2018.

[52] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[53] Yang Chen, Dustin J. Kempton, Azim Ahmadzadeh, Junzhi Wen, Anli Ji, and Rafal A. Angryk. Cgan-based synthetic multivariate time-series generation: a solution to data scarcity in solar flare forecasting. *Neural Comput. Appl.*, 34(16):13339–13353, aug 2022.

[54] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Proceedings of the 30th International*

Conference on Neural Information Processing Systems, NIPS'16, page 2234–2242, Red Hook, NY, USA, 2016. Curran Associates Inc.

[55] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[56] Yang Chen, Dustin J. Kempton, and Rafal A. Angryk. Examining effects of class imbalance on conditional gan training. In *Artificial Intelligence and Soft Computing*, pages 475–486, Cham, 2023. Springer Nature Switzerland.

[57] Sariel Har-Peled and Benjamin Raichel. The fréchet distance revisited and extended. *ACM Trans. Algorithms*, 10(1), jan 2014.

[58] D.C Dowson and B.V Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982.

[59] Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

[60] Elyas Goli, Sagar Vyas, Seid Koric, Nahil Sobh, and Philippe H Geubelle. Chemnet: A deep neural network for advanced composites manufacturing. *The Journal of Physical Chemistry B*, 124(42):9428–9437, 2020.

[61] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[62] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[63] Arnold Benz. Flare observations. *Living Reviews in Solar Physics*, 5, 02 2008.

[64] M. G. Bobra and S. Couvidat. Solar flare prediction using SDO/HMI vector magnetic field data with a machine-learning algorithm. *Astrophys. J.*, 798(2):135, jan 2015.

[65] G. Barnes, K. D. Leka, C. J. Schrijver, T. Colak, R. Qahwaji, O. W. Ashamari, Y. Yuan, J. Zhang, R. T. J. McAteer, D. S. Bloomfield, P. A. Higgins, P. T. Gallagher, D. A. Falconer, M. K. Georgoulis, M. S. Wheatland, C. Balch, T. Dunn, and E. L. Wagner. A comparison of flare forecasting methods. i. results from the "all-clear "workshop. *The Astrophysical Journal*, 829(2):89, Sep 2016.

[66] Chang Liu, Na Deng, Jason TL Wang, and Haimin Wang. Predicting solar flares using sdo/hmi vector magnetic data products and the random forest algorithm. *The Astrophysical Journal*, 843(2):104, 2017.

[67] Monica G Bobra, Xudong Sun, J Todd Hoeksema, M Turmon, Yang Liu, Keiji Hayashi, Graham Barnes, and KD Leka. The helioseismic and magnetic imager (hmi) vector magnetic field pipeline: Sharps–space-weather hmi active region patches. *Solar Physics*, 289(9):3549–3578, 2014.

[68] Rafal Angryk, Petrus Martens, Berkay Aydin, Dustin Kempton, Sushant Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Xumin Cai, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Micheal Schuh, and Manolis Georgoulis. SWAN-SF, 2020.

[69] Atharv Yeoleka, Sagar Patel, Shreejaa Talla, Krishna Rukmini Puthucode, Azim Ahmadzadeh, Viacheslav M Sadykov, and Rafal A Angryk. Feature selection on a flare forecasting testbed: A comparative study of 24 methods. *arXiv preprint arXiv:2109.14770*, 2021.

[70] Mohammad Hossin and MN Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):1, 2015.

[71] A.W. Hanssen and W.J.A. Kuipers. *On the Relationship Between the Frequency of Rain and Various Meteorological Parameters: (with Reference to the Problem Ob Objective Forecasting)*. Koninkl. Nederlands Meterologisch Institut. Mededelingen en Verhandelingen. Staatsdrukkerij- en Uitgeverijbedrijf, 1965.

[72] Christopher C Balch. Updated verification of the space weather prediction center's solar energetic particle prediction model. *Space weather : the international journal of research & applications.*, 6(1), 2008.

[73] J. Brownlee. *Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image Translation*. Machine Learning Mastery, 2019.

[74] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[75] Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P. Bennett. Privacy preserving synthetic health data. *F1000Research*, 8, 2019.

[76] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig

Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[77] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[78] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[79] A. Ben-Hur and J. Weston. A user's guide to support vector machines. *Methods in molecular biology*, 609:223–39, 2010.

[80] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[81] James W. Cooley, Peter A. W. Lewis, and Peter D. Welch. The fast fourier transform

and its applications. *IEEE Transactions on Education*, 12(1):27–34, 1969.

[82] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. *arXiv preprint arXiv:1606.08921*, 2016.

[83] Kyunghyun Cho et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[84] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[85] Lin Ma, Zhuo Chen, Long Xu, and Yihua Yan. Multimodal deep learning for solar radio burst classification. *Pattern Recognition*, 61:573–582, 2017.