On Improving the Performance of Glitch Classification for Gravitational Wave Detection by using Generative Adversarial Networks

Jianqi Yan,1,3 Alex P. Leung,2★ C. Y. Hui,3†
1Faculty of Innovation Engineering, Macau University of Science and Technology, Avenida Wai Long, Taipa, Macau
2Department of Physics, University of Hong Kong, Hong Kong
3Department of Astronomy and Space Science, Chungnam National University, Daejeon 34134, Korea

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

Spectrogram classification plays an important role in analyzing gravitational wave data. In this paper, we propose a framework to improve the classification performance by using Generative Adversarial Networks (GANs). As substantial efforts and expertise are required to annotate spectrograms, the number of training examples is very limited. However, it is well known that deep networks can perform well only when the sample size of the training set is sufficiently large. Furthermore, the imbalanced sample sizes in different classes can also hamper the performance. In order to tackle these problems, we propose a GAN-based data augmentation framework. While standard data augmentation methods for conventional images cannot be applied on spectrograms, we found that a variant of GANs, ProGAN, is capable of generating high-resolution spectrograms which are consistent with the quality of the high-resolution original images and provide a desirable diversity. We have validated our framework by classifying glitches in the Gravity Spy dataset with the GAN-generated spectrograms for training. We show that the proposed method can provide an alternative to transfer learning for the classification of spectrograms using deep networks, i.e. using a high-resolution GAN for data augmentation instead. Furthermore, fluctuations in classification performance with small sample sizes for training and evaluation can be greatly reduced. Using the trained network in our framework, we have also examined the spectrograms with label anomalies in Gravity Spy.

Key words: Methods: data analysis – Techniques: image processing – Gravitational waves

1 INTRODUCTION

Recent state-of-the-art gravitational wave observatories, including Advanced LIGO (Aasi et al. 2015), advanced VIRGO (Acernese et al. 2014) and KAGRA (Akutsu et al. 2020), have brought the possibility of exploring our Universe in a regime beyond the reach of electromagnetic wave. We are now able to investigate the disturbance of spacetime resulted from the merging processes of stellar black holes and neutron stars. This not only allows us to study the nature of these compact objects in a brand-new window, but also enables us to test gravitational theories at the limit of strong gravity.

The basic working principle of a gravitational wave detector is laser interferometry. The arrival of gravitational wave upon the interferometer will lead to a path difference in the lasers running in two orthogonal arms. For a signal with a strain amplitude of $h \lesssim 10^{-21}$ (e.g. Abbott et al. 2016), this causes a path difference smaller than the size of a proton between two arms. All these new possibilities in gravitational wave astronomy rely upon the unprecedented sensitivity of the detectors in identifying such small changes.

Apart from the genuine astrophysical signals, these sensitive detectors also pick up unwanted non-Gaussian noise anomalies which are commonly referred as glitches. The high rate of these glitches can contaminate the gravitational wave data by mimicking or obscuring the genuine astrophysical signals. In order to attain the full potential of the current observatories, it is crucial to identify these noise signals and remove them from the data. Combining efforts from the communities of gravitational physics, human-centered computing, machine learning and citizen science, classification of glitches in LIGO data has been published as the Gravity Spy data (Zevin et al. 2017). Because of the transient nature of glitches, spectrogram is a convenient data representation as it provides a visualization of how does the frequency spectrum of the signal varies with time. The whole Gravity Spy dataset is segregated into 20 different glitch classes according to the morphologies of their spectrograms. Each class corresponds to a different

★ E-mail: alexpl@hku.hk
† Corresponding Author, Email: huichungyue@gmail.com, cyhui@cnu.ac.kr
environmental/engineering origin. For example, the Violin Mode Harmonic class is related to the resonance of the glass fibers that suspend LIGO’s main mirrors.

Since identifying and pinpointing the nature of glitches are important for preventing false detections and improving the instrumental performance, machine learning algorithms have been employed to automate the classification and improve the accuracy (see Section 2.1 for a brief review). In this work, we present a new architecture to improve the glitch classification performance. There are two main parts in our architecture, including data augmentation and classification.

Apart from the small sample size of training data, the imbalanced classes in the Gravity Spy dataset (i.e., the numbers of glitches in different classes vary significantly) can also hamper the classification by deep network. For example, while there are 7476 spectrograms of Bl1p class in the Gravity Spy dataset, we have only 108 spectrograms in the Paired_doves class. For classification with machine learning techniques, imbalanced data often leads to degradation in generalization performance (Sun et al. 2009; Chawla et al. 2004; Mahani & Ali 2019). Furthermore, with small sample sizes, it is well known that deep networks do not generalize well and potentially overfit.

While data augmentation provides a way to tackle the aforementioned problems (Tanaka & Aranha 2019), traditional approaches on image data (e.g., center crop, random crop, flip, rotation, etc.) cannot ensure the consistency and integrity of the information in the spectrogram. For images of celestial objects, such traditional data augmentation methods can exploit certain symmetries (e.g., rotation, translation, reflection) in the data because classification should be invariant to these operations. For example, the classification of a galaxy should not be affected by its orientation, location and size in an image (e.g. Dieleman et al. 2015). However, as each dimension of a spectrogram represents a different quantity (i.e., frequency and time in y and x axes respectively), spectrograms generally lacks the aforementioned symmetries. For example, a translation in the frequency dimension of the Violin Mode Harmonic data can lead to misclassification as its spectrograms appear as short and dot-like at well-defined frequencies (500 Hz and its integral multiples). Therefore, the data augmentation techniques on conventional imaging data cannot be applied to spectrograms.

In this work, we propose a data augmentation method based on Generative Adversarial Networks (GANs) to generate high-resolution spectrograms for training. This method has some successful applications of improving generalizability in the other fields, such as medical imaging (e.g. Sandfort et al. 2019). The community of gravitational wave astronomy has also started employing GANs in their works (e.g., McGinn et al. 2021; Lopez et al. 2022). McGinn et al. (2021) have presented the first application of GANs to explore the space among five classes of simple parametrized waveforms (e.g., sine-Gaussian) to construct new hybrid unmodelled waveforms. They further demonstrated that training the detection algorithm on broader range of signals generated by GANs can enhance the performance. On the other hand, a GAN-based method has recently been utilized to simulate a glitch class of the largest sample size (i.e. Bl1p) in time domain (Lopez et al. 2022), which demonstrates that GANs can generally learn the underlying distribution of this particular class.

Both of the aforementioned works have applied GANs in generating waveforms in time domain. In comparison with the analysis in the time domain, spectrograms have an advantage to better capture the intermittent changes of the signals in both amplitude and frequency. By combining the merits in both time and frequency domains, it enables a visualization for the relationships among the temporal, frequency and amplitude variations in the signal. This provides us with the motivation to explore the capability of generating spectrograms for all known classes of glitches with GANs.

For conventional GANs, they can only generate low-resolution images as they can only handle resolutions at around 128 x 128 and 256 x 256 (Miyato et al. 2018; Liu et al. 2017). A number of different methods of GANs have been considered and experimented. We have found that a variant known as ProGAN, which is recently proposed to generate high-resolution images, works well for our problem with gravitational wave data. Using it to learn in the Gravity Spy dataset, high-resolution spectrograms for each glitch class can be generated with outstanding quality. This provides a way to tackle the imbalanced data problem and improve the overall classification accuracy altogether. Furthermore, our method can avoid overfitting with an augmented training set large and diverse enough for generalization.

This paper is organized as follows: we first give a review on the related works of glitch classification methods and introduce the background for GANs in Section 2. In Section 3, datasets used in this work and the procedures for data preprocessing are described. The details of our proposed framework are given in Section 4. Evaluation criteria and experimental results are presented in Section 5. Finally, a summary and discussions are given in Section 6.

2 BACKGROUND

2.1 Previous Work

Bahaadini et al. (2017) have introduced the “merged-view” model for glitch classification. Each glitch in the Gravity Spy dataset has four spectrograms of different durations (0.5s, 1.0s, 2.0s, and 4.0s). For each event, Bahaadini et al. (2017) combined four \( m \times k \) spectrograms of different duration side by side into a single \( 2m \times 2k \) spectrogram. This approach achieves 96.89% classification accuracy. In later works, ensemble classifier is used to classify the Gravity Spy dataset (Bahaadini et al. 2018). They achieve an accuracy of 98.21% on a standard test set. In their work, the linear support vector machine (SVM) (Mertsalov & McCready 2009), the kernel SVM and the convolutional neural network (CNN) (Lawrence et al. 1997) are employed on the data with all four durations. Their experimental result first indicates that training with 1.0s duration data achieve higher accuracy than with the other durations. Then, with the “merged-view” model, an accuracy of 97.67% was achieved in their work. And by using the ultimate ensemble classifier with hard fusion, an accuracy of 98.21% was reported.

The method of deep transfer learning (Yosinski et al. 2014) has been applied in classifying various astronomical images such as galaxies and stellar clusters (e.g. Domínguez Sánchez et al. 2019; Wei et al. 2020; Khan et al. 2019). For classifying the spectrograms of glitches in the Gravity Spy dataset, George et al. (2017, 2018) have attempted to employ the technique of transfer learning so as to enhance the performance. Instead of concatenating spectrograms as Bahaadini et al. (2017), the authors adopt a new approach of

---

1 Gravity Spy dataset (version 1.0) is used in all of our experiments: https://zenodo.org/record/1476156#.YDBBFQz6aUn
encoding spectrograms with different durations into an RGB image as an input to specially designed CNNs. Using the ResNet50 network architecture (He et al. 2016) and the Inception-V3 network architecture (Szegedy et al. 2016), they have achieved an accuracy of 98.84% on the test set. Another novel structure for transfer learning based on Tsallis entropy on the Gravity Spy dataset is proposed by Ramezani & Pourdarvish (2021). However, this approach only achieves an accuracy of 96.90%.

Although transfer learning apparently improves the performance on a small test set by providing better tuning on the weights, it is uncertain whether it can be well generalized to larger and more diverse data. The key to a successful knowledge transfer depends on two factors: (a) there is a large in common between the source and target domains, and (b) the target learner is not negatively affected by the transferred knowledge (Zhuang et al. 2020). However, as aforementioned, there is a fundamental difference between the source domain (i.e. ImageNet) and the target domain (i.e. spectrograms) in our case. Therefore, the uncertainty whether negative transfer will adversely affect the model generalizability when applied to a larger data set can be a concern. This provides motivation to explore alternative methods for employing deep learning to classify gravitational wave spectrograms.

### 2.2 Generative Adversarial Networks (GANs)

The concept of GANs is proposed by Goodfellow et al. (2014). As an alternative framework, GANs estimate generative models via an adversarial process which includes a generative model $G$ and a discriminative model $D$. The generative model $G$ is used for capturing the data distribution, while the discriminative model $D$ is used to determine whether the sample comes from the data distribution or the model distribution.

GANs use backpropagation and dropout algorithm in training $G$ and $D$. The training of $D$ and $G$ can be represented by playing a two-player min-max game with value function $V(D, G)$ as given by Equation (1):

$$
\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_{z}(z)}[\log(1 - D(G(z)))]
$$

where $p_{data}$ is the generator’s distribution over data $x$, and $p_{z}(z)$ is a prior on input noise variables. A mapping to the data space as $G(z; \theta_G)$, where $G$ is a differentiable function represented by a multilayer perceptron with parameters $\theta_G$ is represented. $D(x)$ represents the probability that $x$ came from the data.

GANs have been demonstrated to be capable in achieving good results on the MNIST (LeCun et al. 1998), Toronto Face Database (TFD) (Susskind et al. 2010), and CIFAR-10 (Krizhevsky et al. 2009) datasets. Shortly thereafter, the conditional generative adversarial nets (CGANs) is presented Mirza & Osindero (2014). They point out that an unconditioned generative model cannot control modes of the generated data. Thus, by conditioning the model on additional conditions, it is possible to direct the data generation process.

Based on retaining the original $G$ and $D$, some additional conditions $y$ (e.g. any class labels or other mode data) are combined with the prior input noise $p_{z}(z)$ in a joint hidden representation in the generator. Similarly, the training process of $G$ and $D$ can be represented by Equation (2):

$$
\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_{z}(z)}[\log(1 - D(G(z|y)))]
$$

Many variants of GANs have been developed in recent years. Such as DCGAN (Radford et al. 2015) and WGAN (Arjovsky et al. 2017). DCGAN combines CNN and GANs, and uses full convolution network and batch normalization (Ioffe & Szegedy 2015). WGAN introduces the Wasserstein distance (also known as the Earth-Mover distance) instead of two similarity measures: the Kullback-Leibler divergence (also known as Relative Entropy) and the Jensen-Shannon divergence. The Wasserstein distance is used to measure the distance between two distributions.

Some variants of GANs are used to generate high-resolution images. A general-purpose framework is proposed to predict pixel by pixel (Isola et al. 2017). It generates new images with 512 × 512 resolutions according to input images. To tackle the difficult problem of generating high-resolution and realistic images by GANs, a new approach that can produce high-resolution images from semantic label maps is proposed by Wang et al. (2018) which generates 2048 × 1024 visually appealing results. The progressive growing of GANs (ProGAN), which can generate high resolution 1024 × 1024 face images, is designed by Karras et al. (2017). Instead of training all layers of the generator and the discriminator as usual, ProGAN gradually grows new layers to generate high-resolution images. In this study, we explore the applicability of ProGAN in improving the performance of spectrogram classification.

### 3 DATA PRE-PROCESSING

Spectrograms in the Gravity Spy dataset are obtained by Q-transform of transient events in the LIGO time series. Each of them is labelled with a unique class (Chatterji et al. 2004). Properties of different classes of glitches in the Gravity Spy dataset are explained in details by Bahaadini et al. (2018). There are 22 glitch classes in Gravity Spy. The names of these classes are mostly self-explanatory (Bahaadini et al. 2018). Among them, the class ‘None_of_the_above’ collects all spectrograms which cannot be unambiguously assigned to the other 21 classes. Also, one should recognize that spectrograms of a number of different classes have rather similar morphologies (e.g. Light_Modulation and Low_frequency_burst), which can possibly lead to misclassification.

For each glitch, spectrograms with four different durations (0.5s, 1.0s, 2.0s, and 4.0s) are included in the Gravity Spy dataset. Each of them covers a frequency range from 10 Hz to 2048 Hz. As an example, we show the spectrograms with different durations of an event of Blip class in Figure 1. Following George et al. (2017), we adopted a method of combining three grey-scale spectrograms with three different durations (1.0s, 2.0s and 4.0s) into a single RGB image. The method is illustrated by Figure 2.

In this work, we have applied the aforementioned data preprocessing on the Gravity Spy dataset (Version 1.0) as in George et al. (2017).

### 4 OUR PROPOSED FRAMEWORK

#### 4.1 Data augmentation

In this work, we use GANs for data augmentation of spectrograms for gravitational wave data. Conventional GANs have difficulties in generating high-resolution spectrograms as they can only handle resolutions at around 128 × 128 and 256 × 256 (Miyato et al. 2018; Liu et al. 2017). We have investigated different variants of GANs for generating high-resolution spectrograms and experimentally found that with ProGAN, encouraging results can be achieved.
Table 1. Different mini-batch configurations for each resolution to minimize the impact of memory constraints in ProGAN.

| mini-batch | 4x4 | 8x8 | 16x16 | 32x32 | 64x64 | 128x128 | 256x256 | 512x512 |
|------------|-----|-----|-------|-------|-------|---------|---------|---------|
| 128        | 128 | 128 | 128   | 64    | 32    | 16      | 8       |

Table 2. The number of each glitch class in the training set, the validation set, and the test set.

| Class Name         | Total (V1.0) | multi-duration (RGB) | training set | validation set | test set |
|--------------------|--------------|----------------------|--------------|----------------|---------|
| 1080 lines         | 1312         | 328                  | 230          | 49             | 49      |
| 1400 ripples        | 928          | 232                  | 162          | 35             | 35      |
| Air_compressor     | 232          | 58                   | 41           | 8              | 9       |
| Blip               | 7476         | 1869                 | 1308         | 281            | 280     |
| Chirp              | 264          | 66                   | 46           | 10             | 10      |
| Extremely_loud     | 1816         | 454                  | 318          | 68             | 68      |
| Helix              | 1116         | 279                  | 195          | 42             | 42      |
| Koi_fish           | 3320         | 830                  | 581          | 125            | 124     |
| Light_modulation   | 2292         | 573                  | 401          | 86             | 86      |
| Low_frequency_burst| 2628         | 657                  | 460          | 99             | 98      |
| Low_frequency_lines| 1812         | 453                  | 317          | 68             | 68      |
| No_glitch          | 724          | 181                  | 127          | 27             | 27      |
| Paired_doves       | 108          | 27                   | 19           | 4              | 4       |
| Power_line         | 1812         | 453                  | 317          | 68             | 68      |
| Repeating_blips    | 1140         | 285                  | 200          | 43             | 42      |
| Scattered_light    | 1836         | 459                  | 321          | 69             | 69      |
| Scratchy           | 1416         | 354                  | 248          | 53             | 53      |
| Tomte             | 464          | 116                  | 81           | 17             | 18      |
| Violin_mode        | 1888         | 472                  | 330          | 71             | 71      |
| Wandering_line     | 176          | 44                   | 31           | 6              | 7       |
| Whistle            | 1220         | 305                  | 213          | 46             | 46      |
| None_of_the_above   | 352          | 88                   | -            | -              | -       |

Figure 1. Example spectrograms for the same event in the Blip class with different durations: (a) 0.5s, (b) 1.0s, (c) 2.0s, and (d) 4.0s.

In the architecture of ProGAN, both generator and discriminator grow progressively: starting from a low resolution (4x4), new layers in a higher resolution (8x8) are then added. Increasingly finer details are modelled as the training progresses with more layers. This procedure is repeated until the resolution reaches 512x512 or 1024x1024. In other words, instead of training all the layers of the generator and the discriminator all at the same time like any other previous GANs approaches, ProGAN gradually increases the number of layers one at a time in order to incorporate high-resolution images.

Figure 2 illustrates the structure of ProGAN. All generated images from ProGAN and real images are used to train the discrimin-
Classification of Gravity Spy data with GANs

The resolution reaches 256x256, the learning rate of the generator is 1.5 x 10^-3. When the last layer of ProGAN is trained, the learning rate of the generator is 3 x 10^-2. As an example, the training process of ProGAN of the Air_Compressor class is shown in Figure 4. We use the same parameters for ProGAN to obtain spectrograms in all 21 classes (excluding the None_of_the_above class).

In our experiments, the 1.0s, 2.0s and 4.0s spectrograms of each glitch event in the Gravity Spy dataset are first converted to grayscale images. Then, those grayscale images are merged into an RGB image by encoding each gray image into red, green, and blue channels, respectively. After all grayscale images are combined to form multi-duration RGB spectrograms, the resultant images are resized to 512x512 to match the input size for ProGAN. The Gravity Spy dataset are split into three subsets randomly: 70% for the training set, 15% for the validation set, and 15% for the test set (See Table 2). In view of their ambiguous nature, spectrograms of the None_of_the_above class are excluded entirely in the training sets, validation sets, and test sets. On the other hand, we will use the optimally trained network in an attempt to classify these spectrograms with unknown nature into the other 21 well-defined classes (See Section 5.3).

A comparison between real images (i.e. the original spectrograms in Gravity Spy dataset) and generated spectrograms by ProGAN is shown in Figure 5. Six classes, including the 1080_LINES class (Row 1), the Air_compressor class (Row 2), the Chirp class (Row 3), the Koi_Fish class (Row 4), the Light_Modulation class (Row 5), and the Repeating_Blips class (Row 6) have been chosen for illustration. In general, ProGAN not only recovers the training data distribution very well, but also increases the diversity of spectrograms which is important for resulting in less overfits and better generalization (See Sect. 4.3 for further details).

Using GANs to generate high-resolution images is often a time-consuming process. The experiment time of training ProGAN are presented by Karras et al. (2017). They trained 1024x1024 images on 8 NVIDIA Tesla V100 GPUs for at least 4 days. All of our experiments are trained on an NVIDIA Tesla V100 (32GB). Each class of the Gravity Spy dataset is trained by 3 days with a fixed epoch separately. However, the image generation process is fast, taking only 5 minutes to generate 1000 512x512 images.

4.2 Classification with deep networks

Many network architectures have been proposed in recent years, including ResNet50 (He et al. 2016), ResNet101 (He et al. 2016), Inception-V3 (Szegedy et al. 2016) and so on. ResNet50 and Inception-V3 have been used for gravitational wave glitch classification in the previous works (George et al. 2017, 2018).

To demonstrate that using GANs as a data augmentation method can improve the spectrogram classification performance, we set up two baselines with: (a) real images (i.e. original spectrograms in Gravity Spy) to train the aforementioned networks without any pre-trained model, and (b) real images to train these networks with a fine-tuned deep transfer learning method. We adopt the pre-train models which are trained with PyTorch on the ImageNet dataset1.

We adopt neural architecture for each network with the number of neurons for the output layer equal to 21 which correspond to the glitch classes defined in Gravity Spy dataset (excluding None_of_the_above). Also, we resize the 512x512 spectrograms to 512x512 resolution to match the input size for ProGAN.

Figure 3. Illustration of the generator and the discriminator of ProGAN. ProGAN generates images with a very low resolution (4 x 4) in the first layer before the second layer is added to the network architecture for a higher resolution (4 x 4 to 8 x 8). This is done recursively. The generator and discriminator always grow in synchrony.

Figure 4. Training process of ProGAN illustrated with an example in the Air_Compressor class. Different epochs of training are shown in a sequential order from left to right and top to bottom. One can notice the progressive increase in resolution.

2 https://github.com/tkarras/progressive_growing_of_gans

3 https://pytorch.org/vision/stable/models.html
Figure 5. Comparison between original Gravity Spy spectrograms (Column 1) and the GAN-generated data (Column 2, Column 3, Column 4).
for fitting the specific size of input layer of a particular network architecture. We adopt an input dimension of 299×299 for Inception-V3 and 244×244 for all the other networks respectively.

The early-stopping method and the L2-regularization are used to prevent the training process from overfitting. In each training epoch, the loss of training set and validation set will be recorded. Once the loss of the validation set is greater than the last record for five consecutive times, the training will stop. The dynamic adjustment strategy of the learning rate is used in the training process with an initial learning rate of 10^{-4}.

For resolving the imbalanced data problem, we use ProGAN to generate the spectrograms for different class with Algorithm 1. For each class, if \( N_{\text{real}} \) real spectrograms is less than a predefined number \( N_{\text{total}} \), we will generate \( N_{\text{total}} - N_{\text{real}} \) images for compensation. We combine the generated images and the original Gravity Spy spectrograms for the training set of the classification network. In our experiments, we have tried different values of \( N_{\text{total}} \) which is taken to be 500, 2000, 3000, and 5000. We would like to examine how does the classification performance vary with this parameter.

**Algorithm 1: Data Augmentation Algorithm**

Input: \( N_{\text{real}} \) real images in one class

Output: \( N_{\text{aug}} \) generated images using ProGAN in each class

for each class do
    if \( N_{\text{real}} < N_{\text{total}} \) then
        \( N_{\text{aug}} = N_{\text{total}} - N_{\text{real}} \)
    else
        \( N_{\text{aug}} = 0 \)
    end
end
Return \( N_{\text{aug}} \) generated images

Our experimental procedures are illustrated in Figure 6. Prior to splitting the Gravity Spy dataset, the blank spectrograms in each glitch class are removed. The dataset is split into three subsets: 70% for the training set, 15% for the validation set, and 15% for the test set. The validation set is used to tune hyperparameters of the network and the test set is used to evaluate the classification performance. The augmented data by GANs with Algorithm 1 are only added in the training set. Finally, the confusion matrix and the weighted F1-scores are used to evaluate the glitch classification performance.

In order to evaluate the fluctuations of classification performance, we run each experiment five times. During the training process, different layers in network architecture are trained with different random operations (e.g. random weights initialization for the experiments without a pre-trained model, batch normalization, dropout, data shuffling). On the other hand, we kept the same hyperparameters initialization (e.g. initial learning rate, batch size, etc.) for all the experiments. And for those with pre-training incorporated, the weight initialization always starts with the same pre-trained weights in each set of experiments.

### 4.3 Performance metrics for GANs

For evaluating the performance of GANs (i.e. quality of the generated images), there are many effective performance metrics. The Inception Score, proposed by Salimans et al. (2016), measures the quality of a generated image by computing the KL-divergence between the conditional and marginal probability distributions. The Inception Score seeks to capture the image quality and the image diversity of a collection of generated images. ProGAN achieves a record Inception Score of 8.80 in unsupervised CIFAR10 (Karras et al. 2017).

However, limitations of the Inception Score is mentioned by Barratt & Sharma (2018). The authors suggest that the Inception Score is only a good measure for the ImageNet dataset. Furthermore, the Inception Score is sensitive to training weights. The scores can fluctuate in response to the small changes of the training weights in the same deep learning architecture.

Recently, a novel evaluation method of GANs is proposed by Shmelkov et al. (2018), which includes two scores: GAN-train and GAN-test. While GAN-train learns a classifier on GAN-generated images and measures the performance on real test images, GAN-test measures how realistic GAN-generated images are. The procedures are illustrated in Figure 7. The diversity and the consistency with the original data of GAN-generated images are evaluated by GAN-train. When the GAN-train score on test set is close to that on the validation set, it shows that the diversity of generated images is desirable. On the other hand, GAN-test learns a classifier on real images and evaluates this classifier on GAN-generated images. A high GAN-test means that the quality of generated images is close to the quality of real images. Ideally, GAN-test score on the test set should be close to the score of the validation set.

Both GAN-train score and GAN-test score are used to evaluate the performance of ProGAN in our experiments. In this work, spectrograms in the Gravity Spy dataset are randomly split into three subsets: 70% for the training set, 15% for the validation set, and 15% for the test set. We generate 10000 spectrograms for each class. We use the training set to train a classifier and evaluate the GAN-test score by using the generated images as the test set in this experiment. Similarly, we use the generated images to train a new classifier and evaluate the GAN-train score by using the test set. In this experiment, the network architectures of classifiers are ResNet50, ResNet101 and Inception-v3.

### 5 EXPERIMENTAL RESULTS

#### 5.1 Evaluation of GANs performance in spectrogram generations

With 10000 generated images for each class by ProGAN, the performance of different network architectures is compared by using GAN-train and GAN-test scores in Table 3. In our experiment with multi-duration RGB spectrograms, both GAN-train and GAN-test scores are desirable. The highest GAN-train and GAN-test are 98.43% and 99.35%, respectively. This implies that the multi-duration spectrograms generated by GANs are sufficiently diverse and realistic. Overall, based on these performance metrics, we conclude that ProGAN is capable to generate images with an acceptable quality and diversity with the multi-duration RGB spectrograms.

In order to further examine the consistency between GAN-
generated images and real images, we use the backbone of the Inception-V3 network architecture for feature extraction and t-SNE (Van der Maaten & Hinton 2008) for unsupervised learning. To validate the effectiveness of our GAN-based framework, we use the backbone of the optimal Inception-V3 network architecture \(N_{total} = 5000\) (Without Pre-training cf. Table 4) and remove its final fully connected layer. Therefore, for all multi-durations RGB spectrogram inputs, the trained model outputs a high dimensional feature vector and this vector is used as input for t-SNE. We used t-SNE to reduce the high-dimensional feature vectors into a 3D feature space for visualization. 10 real images and 10 GAN-generated images in each class are randomly selected for the experiments.

Figure 8 shows the clustering results of the t-SNE algorithm. Different colours represent different types of glitches. We found that the real images (circles in Figure 8) and the generated images (triangles in Figure 8) from the same class are clustered in a similar way in this reduced 3D feature space.

We notice that a generated image from the Violin_Mode class is clustered with those from the Whistle class (see the bottom panel of Figure 8). This may account for a misclassification between these two classes as seen in the confusion matrix shown in Figure 10. Except for this, we found the GAN-generated data are generally consistent with the original Gravity Spy data.

For testing the robustness of the clustering results by t-SNE, we have run the experiment with 7 types of glitches for 9 times. We have included 1080Lines, 140Ripples, Air_Compressor, Blip, Chirp, Extremely_Loud and Helix in this robustness test. Different random seeds were set in each run. The results are shown in Appendix A1. Both real and generated images are found to be clustered closely and different classes are well differentiated in each...
5.2 Classification of Gravity Spy data

Top-1 accuracy is the proportion of correctly classified images with their true classes equal to the predict classes with the highest probabilities (i.e. the conventional definition of accuracy), which is the common evaluation criteria to gauge the classification performance in image classification (Simonyan & Zisserman 2014). However, top-1 accuracy is most used in balanced data when all the classes are equally important. On the other hand, there are many works using F1-score to evaluate the classification performance (e.g. Li et al. 2017; Sullivan et al. 2018). F1-score is based on the harmonic mean of precision and recall. The most common F1-scores include the micro-averaged F1-score, the macro-averaged F1-score and the weighted F1-score. While micro-averaged F1-score is obtained by computing total precision and recall globally, macro-averaged F1-score is the unweighted average of the precision and recall for each class.

The weighted F1-score is a variant of macro-averaged F1-score. The key difference between the weighted F1-score and the macro-averaged F1-score is how precision and recall are computed for each class. According to the sample size of the test set and the number of true images of each class, weights are determined for each class and the weighted F1-score is calculated according to the weighted precision and the weighted recall for each class. In view of the imbalanced classes in the test set, we decided to use the weighted F1-score instead of the top-1 accuracy to evaluate the classification performance in this work. In addition, we also used the confusion matrix to visualize the performance in our experiments.

In order to increase the number of spectrograms in each class in the training data and tackle the imbalanced data problem, we mixed the spectrograms generated by ProGAN with the real images from the Gravity Spy data for training (cf. algorithm 1). In Table 4, we compare the weighted F1-score of different $N_{total}$ on different network architectures. We have run each experiment five times for
Once the baselines have been set, we performed a series of experiments with different \( N_{\text{total}} \) for comparisons. The results are summarized in Table 4 and Figure 9. The quoted error bars correspond to the maximum variations of weighted F1-scores. For all the network architectures considered in this work, the GAN-based data augmentation method can generally lead to more accurate classification with respect to the baselines without pre-training. In the case of Inception-V3, it achieves the highest average weighted F1-score of 0.9885 for \( N_{\text{total}} = 5000 \) without pre-training which is higher than both corresponding baselines. This shows that our method can lead to a performance improvement comparable or even exceed those attained by transfer-learning. For the experiments which incorporated both pre-training and data augmentation, the highest average weighted F1-score of 0.9891 is achieved by the Inception-V3 network architecture with \( N_{\text{total}} = 2000 \) and pre-training.

We have also investigated the impact of GAN-based data augmentation on recognizing the classes with small sample size. We focus on the classes Air_Compressor, Paired_Doves and Wandering_Line, in the Gravity Spy dataset. In this part, we only adopted the Inception-V3 network architecture because of its superior performance as it allows a larger input size. In Table 5, apart from the averaged weighted F1-scores (Column 8) and the standard deviations (Column 9), the weighted F1-scores in each five runs of all the experiments are also explicitly shown in Columns 3-7.

The baseline performances are given in the first two rows of Table 5. Without data augmentation and pre-training, the classification performance on the small classes apparently suffer from significant fluctuations. The largest impact is found on the class with the smallest sample size (i.e. Paired_Doves), which has the weighted F1-score varied from 0.667 to 1 in our experiments. On the other hand, for the baseline with pre-training incorporated, the fluctuations are reduced considerably (e.g. perfect precision and recall on Paired_Doves are achieved).

To examine the influence of our framework, we incorporated data augmentation by varying \( N_{\text{total}} \) without any pre-training. The results are summarized in Rows 3-6 of Table 5. We found that the fluctuations in all three classes can generally be reduced by our method. For the case of Paired_Doves, we obtained perfect precision and recall even with \( N_{\text{total}} = 500 \). For Wandering_Line, we have achieved an average weighted F1-score of 0.987 with \( N_{\text{total}} = 5000 \) which exceeds that attained by the baseline with pre-training.

For the case of Air_Compressor, we notice that in all the experiments with pre-training and/or GAN-based data augmentation incorporated, the weighted F1-score kept fluctuating between two values: 0.941 and 1. Examining the confusion matrices found that one of the nine samples in this test set is frequently mis-classified in the class of Power_Line. This prompts us to inspect its test data visually. We found that the spectrogram of this mis-classified data
Table 4. The average weighted F1 scores of glitch classification on the multi-duration test set from the Gravity Spy dataset. The maximum fluctuations among five runs in each experiment are shown.

| Model                        | Without pre-training | With pre-training | Without data augmentation | With data augmentation only | With both data augmentation and pre-training |
|------------------------------|----------------------|-------------------|---------------------------|-----------------------------|---------------------------------------------|
| ResNet50 (224×224)           | 0.9811 ±0.0015       | 0.9876 ±0.0013    | 0.9812 ±0.0042            | 0.9834 ±0.0025              | 0.9880 ±0.0018                            |
| ResNet101 (224×224)          | 0.9884 ±0.0021       | 0.9832 ±0.0028    | 0.9864 ±0.0031            | 0.9863 ±0.0045              | 0.9888 ±0.0015                            |
| Inception-V3 (299×299)       | 0.9854 ±0.0020       | 0.9858 ±0.0026    | 0.9871 ±0.0027            | 0.9891 ±0.0032              | 0.9891 ±0.0016                            |

Figure 10. Confusion matrix with experimental results from the Inception-V3 network architecture (N_total = 5000, without pre-training).
5.3 Classifying spectrograms from None_of_the_Above

The class None_of_the_Above collects all the glitches that cannot be put into the other 21 classes without ambiguity. Since the spectrograms in this class are unlikely having a common origin, all the data in this class were ignored in all the aforementioned experiments. With the improved performance by using GAN-based data augmentation, we now attempt to classify the None_of_the_Above spectrograms into the other 21 well-defined classes.

![Figure 11. Confusion matrix with experimental results from the Inception-V3 network architecture (N_{total} = 2000, with pre-training).](image)

![Figure 12. Predicted classes on the None_of_the_Above spectrograms with two different network architectures. For each network design, results from five individual model predictions, soft-voting and hard-voting are shown.](image)
Table 5. The weighted F1-scores for classifying the multi-duration test set of three classes with small sample sizes from the Gravity Spy dataset.

| Class               | 1       | 2       | 3       | 4       | 5       | Average Weighted F1 Score | Standard Deviation |
|---------------------|---------|---------|---------|---------|---------|---------------------------|--------------------|
| **Baselines (i.e. without data augmentation)** |         |         |         |         |         |                          |                    |
| Without Pre-training |         |         |         |         |         |                          |                    |
| Air_Compressor      | 0.941   | 1       | 1       | 1       | 1       | 0.988                     | 0.026              |
| Paired_Doves        | 0.667   | 0.667   | 1       | 0.667   | 0.667   | 0.733                     | 0.149              |
| Wandering_Line      | 0.833   | 0.727   | 1       | 0.714   | 0.923   | 0.840                     | 0.124              |
| Pre-training        |         |         |         |         |         |                          |                    |
| Air_Compressor      | 1       | 1       | 1       | 1       | 0.941   | 0.988                     | 0.026              |
| Paired_Doves        | 1       | 1       | 1       | 1       | 1       | 0.985                     | 0.034              |
| Wandering_Line      | 1       | 1       | 0.923   | 1       | 0.985   | 0.985                     | 0.030              |
| With data augmentation only |         |         |         |         |         |                          |                    |
| 500 (Without Pre-training) |         |         |         |         |         |                          |                    |
| Air_Compressor      | 1       | 1       | 1       | 0.941   | 0.941   | 0.978                     | 0.031              |
| Paired_Doves        | 1       | 1       | 1       | 1       | 1       | 0.985                     | 0.063              |
| Wandering_Line      | 0.875   | 1       | 1       | 0.875   | 0.933   | 0.937                     | 0.032              |
| 2000 (Without Pre-training) |         |         |         |         |         |                          |                    |
| Air_Compressor      | 1       | 1       | 1       | 1       | 0.941   | 0.976                     | 0.032              |
| Paired_Doves        | 1       | 1       | 1       | 1       | 1       | 0.985                     | 0.032              |
| Wandering_Line      | 0.933   | 1       | 1       | 0.933   | 0.923   | 0.933                     | 0.045              |
| 3000 (Without Pre-training) |         |         |         |         |         |                          |                    |
| Air_Compressor      | 0.941   | 1       | 0.941   | 0.941   | 1       | 0.953                     | 0.026              |
| Paired_Doves        | 1       | 1       | 1       | 1       | 1       | 0.987                     | 0.034              |
| Wandering_Line      | 1       | 0.857   | 0.857   | 1       | 0.943   | 0.987                     | 0.078              |
| 5000 (Without Pre-training) |         |         |         |         |         |                          |                    |
| Air_Compressor      | 0.941   | 1       | 0.941   | 0.941   | 0.941   | 0.953                     | 0.026              |
| Paired_Doves        | 1       | 1       | 1       | 1       | 1       | 0.987                     | 0.030              |
| Wandering_Line      | 0.923   | 0.923   | 0.923   | 1       | 0.933   | 0.987                     | 0.042              |

and adopted $N_{total} = 500$). The reason of considering two network architectures is for cross-checking whether consistent predictions can be resulted from different network design.

Many methods are used for quantifying uncertainty in deep learning, such as Dropout (Srivastava et al. 2014; Gal & Ghahramani 2016) and ensemble averaging methods (Lakshminarayanan et al. 2017; Mani & Tsai 2017). Since there is no Dropout layer in the ResNet50 network architecture (He et al. 2016), we employed ensemble averaging method to quantify the uncertainty of the predictions in this experiment. Similar to the other experiments in this work (see. Sec. 5.2), we repeat the prediction five times from each model and used both soft voting and hard voting for obtaining the final results from each network architecture. The distributions of the predicted classes from both networks are summarized in Figure 12. We show that different architecture designs and voting schemes lead to consistent results in this experiment.
To gain further confidence on the predictions, we have also visually compared the spectrograms from None_of_the_Above with those in the corresponding classes that our models have predicted. In Figure 13, we compare a few None_of_the_Above spectrograms with those from the classes which they are assigned by our model. One should note the similarity between these unidentified spectrograms with the confirmed samples in their corresponding predicted classes.

Figure 13. Comparison of some spectrograms in the None_of_the_Above class and some well-defined examples in the corresponding classes. The predictions of those spectrograms are assigned by our model. Designations of the selected data in Gravity Spy are given below each spectrogram.

6 SUMMARY & DISCUSSION

In this work, we have demonstrated that the classification of spectrograms can be improved by incorporating GANs. A variant of GANs, ProGAN, is adopted in this experiment for generating high-resolution spectrograms. Here we summarized our major results:

(i) While conventional methods of data augmentation is not applicable in spectrogram classification, we have demonstrated that ProGAN not only provides a framework to generate spectrograms that are consistent with the real data (see Figure 5), but also increases the diversity which can result in a better generalization (Table 3). Using the high-resolution spectrograms generated by ProGAN, we enlarged the training data so as to uplifting both problems of small sample and imbalanced classes in the original data (cf. Algorithm 1).

(ii) We have shown that with the aid of GAN-base data augmentation, we can attain a performance of deep learning classification comparable with that obtained by the pre-trained model (Table 4 and Figure 9). This suggests that our framework can provide an alternative method of employing deep learning to classify gravitational wave spectrograms without using transfer-learning.

(iii) We found that the impact of our framework is particularly significant on the classes with small sample size. By including the GAN-generated spectrograms for training, the fluctuations in classifying the small classes can be greatly reduced (Table 5).

Besides the glitch classification considered in this work, our method can be applied to a wider range of spectrograms obtained from the gravitational wave detectors. Different origins of gravitational wave signals show different patterns in the spectrograms. For example, a core collapsed supernova (CCSN) can trigger a number of possible features in the associated gravitational wave signals, e.g. a low-order g-mode from the proto-neutron star surface oscillation has its frequency grows in a timescale $\sim 1$ s as it contracts. Such feature can be visualized as a defining pattern in the spectrograms. There is a general consensus that the outward-going shock will be stalled shortly after the bounce. And it requires the neutrino heating from the proto-neutron star to re-accelerate the shock and drive the prompt explosion. This can trigger the standing accretion shock instability (SASI) which can generate the gravitational wave signal as a distinguishable feature in the spectrograms (e.g see Fig. 9 & 10 in Pan et al. 2021).

While the technique of matched filtering is found to be successful in uncovering the gravitational wave bursts from the compact
binary coalescences which have well-defined waveforms as determined by numerical relativity, such method may have difficulties in searching the signals from CCSNe. Since the gravitational wave produced by CCSNe are expected to be affected by the turbulence, it should be stochastic in nature and hence the waveform cannot be deterministically predicted (Kotake et al. 2009; Aloy et al. 2019).

Recently, the feasibility of detecting CCSN gravitational wave signals by deep learning has been considered (e.g. Chan et al. 2020). However, because of the 3-D simulations of CCSNe are computationally expensive, the simulated data for training are suffered from the same problems that we have considered in this work (i.e small sample size and imbalanced classes, see Table II in Chan et al. 2020, for examples). By incorporating GAN-based data augmentation suggested in our work, not only the aforementioned problems can be relieved, the diversity of the training data can also be increased. Owing to the stochastic nature of the signals, the actual waveform from the real event is likely to show features which are unexpected from the simulations (Chan et al. 2020). By increasing the diversity of the training data and hence enlarging the parameter space coverage, GANs can potentially enhance the detectability of such signals.

ACKNOWLEDGEMENTS

CYH is supported by the research fund of Chungnam National University. The work of JY is also supported by the National Research Foundation of Korea through grants 2016R1A5A1013277 and 2022R1F1A1073952. In addition, the work of Jianqi Yan and Alex Po Leung is supported by the Science and Technology Development Fund, Macau SAR (No. 0079/2019/A2).

DATA AVAILABILITY

The data underlying this article are available in Zenodo, at https://zenodo.org/record/1476156#.YUSN6bgzalh. DOI: https://dx.doi.org/10.5281/zenodo.1476156.

REFERENCES

Aasi J., et al., 2015, Classical and quantum gravity, 32, 074001
Abbott B. P., et al., 2016, Phys. Rev. Lett., 116, 061102
Acernese F., et al., 2014, Classical and Quantum Gravity, 32, 024001
Akutsu T., et al., 2020, Progress of Theoretical and Experimental Physics
Aloy M. A., Ibáñez J. M., Sanchis-Gual N., Obergaulinger M., Font J. A., Akutsu T., et al., 2020, Progress of Theoretical and Experimental Physics
Abbott B. P., et al., 2016, Phys. Rev. Lett., 116, 061102
Aasi J., et al., 2015, Classical and Quantum Gravity, 32, 074001
Akiyama T., et al., 2019, Monthly Notices of the Royal Astronomical Society, 494, 4890
Arjovsky M., Chintala S., Bottou L., 2017, in International conference on machine learning, pp 214–223
Bahadur S., Rohani N., Coughlin S., Zevin M., Kalogera V., Katsaggelos A. K., 2017, in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 2931–2935
Bahadur S., Noroozi V., Rohani N., Coughlin S., Zevin M., Smith J. R., Kalogera V., Katsaggelos A., 2018, Information Sciences, 444, 172
Barratt S., Sharma R., 2018, arXiv preprint arXiv:1801.01973
Chan M. L., Heng I. S., Messenger C., 2020, Phys. Rev. D, 102, 043022
Chatterji S., Blackburn L., Martin G., Katsavounidis E., 2004, Classical and Quantum Gravity, 21, S1809
Chlawa N. V., Jakpowicz N., Kotcz A., 2004, ACM SIGKDD explorations newsletter, 6, 1
Dielemann S., Willott K. W., Dambre J., 2015, Monthly Notices of the Royal Astronomical Society, 450, 1441
Domínguez Sánchez H., et al., 2019, Monthly Notices of the Royal Astronomical Society, 484, 93
Gal Y., Gharahmami Z., 2016, in international conference on machine learning, pp 1050–1059
George D., Shen H., Huerta E., 2017, arXiv preprint arXiv:1706.07446
George D., Shen H., Huerta E., 2018, Physical Review D, 97, 101501
Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., 2014, arXiv preprint arXiv:1406.2661
He K., Zhang X., Ren S., Sun J., 2016, in Proceedings of the IEEE conference on computer vision and pattern recognition. pp 770–778
Ioffe S., Szegedy C., 2015, in International conference on machine learning. pp 448–456
Isola P., Zhu J.-Y., Zhou T., Efros A. A., 2017, in Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1125–1134
Karras T., Aila T., Laine S., Lehuiten J., 2017, arXiv preprint arXiv:1710.10196
Khan A., Huerta E., Wang S., Gruendl R., Jennings E., Zheng H., 2019, Physics Letters B, 795, 248
Kotake K., Iwakami W., Ohnishi N., Yamada S., 2009, The Astrophysical Journal, 697, L133
Krizhevsky A., Hinton G., et al., 2009
Lakshminarayanan B., Pritzel A., Blundell C., 2017, in Advances in neural information processing systems, 30
Lawrence S., Giles C. L., Tsai F. A., Back A. D., 1997, IEEE transactions on neural networks, 8, 98
LeCun Y., Bottou L., Bengio Y., Haffner P., 1998, Proceedings of the IEEE, 86, 2278
Li Y., Song Y., Luo J., 2017, in Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3617–3625
Liu M.-m., Zhang M.-q., Liu J., Zhang Y.-n., Ke Y., 2017, arXiv preprint arXiv:1712.06951
Lopez M., Boudart V., Bujsmans K., Reza A., Caudill S., 2022, arXiv preprint arXiv:2203.06494
Mahani A., Ali A. R. B., 2019, Recent Trends in Computational Intelligence, pp 1–23
Mami A., Tsai F.-T., Chen J., 2017, Journal of Hydrologic Engineering, 22, 04016067
McGinn J., Messenger C., Williams M. J., Heng I. S., 2021, Classical and Quantum Gravity, 38, 155005
Mertsalov K., McCreary M., 2009, ACM Computing Surveys (CSUR), pp 1–47
Mirza M., Osindero S., 2014, arXiv preprint arXiv:1411.1784
Miyato T., Kataoka T., Koyama M., Yoshida T., 2018, arXiv preprint arXiv:1802.05957
Pan K.-C., Liebendorfer M., Couch S. M., Thielemann F.-K., 2021, The Astrophysical Journal, 914, 140
Radford A., Metz L., Chintala S., 2015, arXiv preprint arXiv:1511.06434
Ramezani Z., Pourdarvish A., 2021, arXiv preprint arXiv:2203.06494
Ramezani Z., Pourdarvish A., 2021, arXiv preprint arXiv:1802.05957
Shmelkov K., Schmid C., Alahari K., 2018, in Proceedings of the European Conference on Computer Vision (ECCV). pp 213–229
Simonov N., Zisserman K., 2014, arXiv preprint arXiv:1409.1556
Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014, The journal of machine learning research, 15, 1929
Sullivan D. P., et al., 2018, Nature biotechnology, 36, 820
Sun Y., Wong A. K., Kamel M. S., 2009, International journal of pattern recognition and artificial intelligence, 23, 687
Susskind J., Anderson A., Hinton G. E., 2010, Technical report, The Toronto face dataset. Technical Report UTML TR 2010-001, U. Toronto
Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z., 2016, in Proceedings of the IEEE conference on computer vision and pattern recognition. pp 2818–2826
Takana F. H. K. d. S., Aranha C., 2019, arXiv preprint arXiv:1904.09135
Van der Maaten L., Hinton G., 2008, Journal of machine learning research.
Figure A1. t-SNE experimental results with different random seeds. Seven types of glitches in the experiment are selected.

APPENDIX A: ROBUSTNESS TEST OF T-SNE EXPERIMENTS WITH DIFFERENT RANDOM SEEDS

This paper has been typeset from a TeX/\LaTeX\ file prepared by the author.