Simulating transient noise bursts in LIGO with generative adversarial networks

Melissa Lopez, Vincent Boudart, Kerwin Buijsman, Amit Reza, and Sarah Caudill

Phys. Rev. D **106**, 023027 — Published 26 July 2022

DOI: 10.1103/PhysRevD.106.023027

Simulating Transient Noise Bursts in LIGO with Generative Adversarial Networks

Melissa Lopez,[1,2,a] Vincent Boudart,[3] Kerwin Buijsman,[4,5,b] Amit Reza,[1,2] and Sarah Caudill[1,2]

[1] *Institute for Gravitational and Subatomic Physics (GRASP), Department of Physics, Utrecht University, Princetonplein 1, 3584 CC Utrecht, Netherlands*
[2] *Nikhef, Science Park 105, 1098 XG Amsterdam, Netherlands*
[3] *STAR Institute, Bâtiment B5, Université de Liège, Sart Tilman B4000 Liège, Belgium*
[4] *Randstad, Diemermere 25, 1112TC Diemen, Netherlands*
[5] *Gravitation and Astroparticle Physics Amsterdam (GRAPPA), Institute for Theoretical Physics Amsterdam, University of Amsterdam, the Netherlands*

(Dated: June 13, 2022)

The noise of gravitational-wave (GW) interferometers limits their sensitivity and impacts the data quality, hindering the detection of GW signals from astrophysical sources. For transient searches, the most problematic are transient noise artifacts, known as glitches, that happen at a rate around 1 min^{-1}, and can mimic GW signals. Because of this, there is a need for better modeling and inclusion of glitches in large-scale studies, such as stress testing the pipelines. In this proof-of concept work we employ Generative Adversarial Networks (GAN), a state-of-the-art Deep Learning algorithm inspired by Game Theory, to learn the underlying distribution of blip glitches and to generate artificial populations. We reconstruct the glitch in the time-domain, providing a smooth input that the GAN can learn. With this methodology, we can create distributions of $\sim 10^3$ glitches from Hanford and Livingston detectors in less than one second. Furthermore, we employ several metrics to measure the performance of our methodology and the quality of its generations. This investigation will be extended in the future to different glitch classes with the final goal of creating an open-source interface for mock data generation.

I. INTRODUCTION

During the first observing run (O1), the existence of gravitational-wave (GW) signal from binary black hole (BBH) coalescence was successfully proven by Advanced Laser Interferometer Gravitational-Wave Observatory (LIGO) [1]. After an upgrade of the detectors to increase their sensitivity, Advanced LIGO [2] started in November 2016 the second observing run (O2), which Advanced Virgo [3] joined in August 2017 [4]. Following significant upgrades, in April 2019, the third observing run (O3) was initiated by Advanced LIGO, and Advanced Virgo [5]. During O1 and O2, 11 candidates were detected and 74 were detected during O3 [6–8]. In the coming years, the improvement of the second generation of interferometers and the construction of the third generation of detectors, such as Cosmic Explorer, LISA, and Einstein Telescope, will increase significantly the detection sensitivity [9–11].

While current GW search techniques for transient signals ($\lesssim$ 1 minute) have been extremely successful, their sensitivity continues to be hindered by the presence of transient bursts of non-Gaussian noise in the detectors, known as *glitches*. Glitches have durations typically on the order of sub-seconds, and their causes can be environmental (e.g., earthquakes, wind, anthropogenic noise) or instrumental (e.g., overflows, scattered light [12]), although in many cases, the cause remains unknown [13]. While much work has been done to mitigate the effect of glitches on GW searches [14, 15], they remain one of the major limiting factors in the detection and parameter estimation of transient GW signals.

In this paper, we learn the underlying distribution of glitches with Machine Learning (ML) methods for better modelling an inclusion for large scales. For this aim, we employ Generative Adversarial Networks (GAN) [16] to build an artificial population of glitches and we use several metrics to test their similarity to the real input. This paper is structured as follows. In section II we introduce the current state-of-the-art of glitch identification, as well as blip glitches, which is the focus of this work. In section III we describe in detail the ML method employed and we give details about the data acquisition. In section IV we present some examples of the generated data, we propose several statistical tests to measure the performance of our methodology and we comment on its limitations. In section V we provide a description of several possible applications of the generated data for future investigations and in section VI we conclude.

II. GRAVITATIONAL-WAVE DETECTOR GLITCHES

A. Identification and Classification

Because glitches can reduce the amount of analyzable data, bias astrophysical detection, parameter estimation, and even mimic GW signals, it is fundamental to develop robust techniques to identify and characterize these sources of noise for their possible elimination. In previ-

[a] Corresponding author: m.lopez@uu.nl
[b] This research was conducted during my studies at the University of Amsterdam.

ous LIGO and Virgo science runs, this classification was performed by visual inspection, which soon proved to be slow and inefficient [17].

During O2 run, the detection rate of glitches was $\approx 1\,\mathrm{min}^{-1}$; so due to the overwhelming amount of glitches present in data. A promising option is to construct ML algorithms to identify and classify glitches [17–19]. However, another challenge arises since a pre-labeled data set is necessary to train such algorithms. With this goal in mind, Zevin et al. [20] developed pioneerwork to classify transient noise, called *Gravity Spy*. In this work, both problems are addressed: volunteers provide large labeled data sets to train the ML algorithms through Zooniverse infrastructure, while ML algorithms learn to classify the rest of the glitches correctly, providing feedback to participants. In practice, a glitch time series that we wish to classify is fed to the algorithm that generates the Q-transform of its input (see [20] for details). Then, *Gravity Spy* classifier assigns a class and a confidence value c_{GS} to the Q-transform of the glitch, where c_{GS} represents the confidence of the label assigned. *Gravity Spy* uses a multi-class classification, and it differentiates between 23 glitch classes and the absence of glitch inside the Q-scan in O2 [20].

B. Blip Glitches

This work focuses on blip glitches due to their abundance during O2 run and their simple morphology. Blip glitches are short glitches ($\lesssim 0.2$ s) that have a characteristic morphology of a symmetric 'teardrop' shape in time-frequency in the range $[30, 250]$ Hz, as we show in Fig.1 (left). They appear in both Livingston (L1) and Hanford (H1) detectors, which is the focus of our work, but there is also evidence of their presence in Virgo and GEO 600 [13]. Due to their abundance and form, blip glitches hinder both the unmodeled burst and modeled CBC searches [21, 22], with particular emphasis in compact binaries with large total mass, highly asymmetric component masses, and spins anti-aligned with the orbital angular momentum. For illustration, in Fig.1, we can observe the similarities between a blip an intermediate binary black-hole chirp surrounded by O2 noise. Moreover, since there is no clear correlation to the auxiliary channels, they cannot be removed from astrophysical searches yet.

III. METHODOLOGY

ML techniques have been very successfully applied for solving a variety of tasks across different domains, and in recent times they have sparked the interest of scientists in the field of GW data analysis. A widely used ML method for pattern recognition is convolutional neural networks (CNNs), which present a grid-like topology, able to exhibit strong local spatial dependencies, allowing faster evaluation speeds [23]. CNNs has been successfully employed in different tasks such as identification of BBH [24, 25] and binary neutron stars (BNS) [26, 27], detection of the early inspiral of BNS mergers [28, 29], supernovae identification [30–32] and glitch classification [20, 33], among others. See also [34] for an interesting review. CNNs can also be used to achieve pixel-wise identification of long-duration bursts in the time-frequency plane. Indeed, authors in [35] built a network that learns to identify the relevant pixels in the image to later use this information to upsample [36] it into the original size.

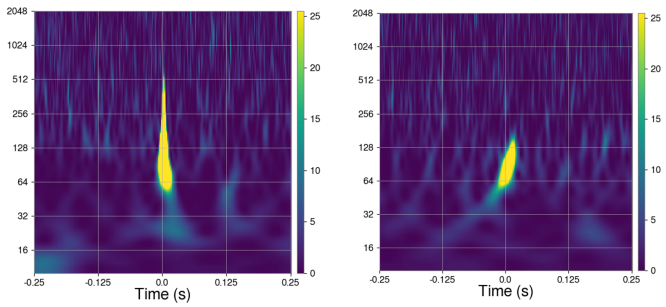


FIG. 1. *(Left)* Q-transform of a blip glitch retrieved from *Gravity Spy* [20]. *(Right)* Q-transform of an event with total mass $106.6^{+13.5}_{-14.8} M_{\odot}$.

ML methods are not only limited to pattern recognition tasks. GAN can learn the underlying distribution of a population to produce artificial examples from Gaussian noise. With this idea in mind, the authors in [37] employed a conditional GAN to burst signals, allowing them to generate multiple classes of signals with the same algorithm and to interpolate through different classes, creating mixed signals. The powerful generation capability of GAN leads the authors to foresee that it could be applied to generate artificial glitches. In the following subsection, we provide more details about GAN methodology and the architecture of our network.

A. Generative Adversarial Networks

GAN [16] are a class of generative algorithms in which two neural networks compete with each other to achieve realistic image generation. One network, known as the *generator*, is responsible for generating new images from random noise, while the other, known as the *discriminator*, tries to discriminate the generated images from the real training data. The generator progressively learns which features of the real images should be mimicked to fool the discriminator and save them into the latent space, which can be understood as a compressed representation of the input data learnt by the generator. At the end of the training, new images are drawn by randomly taking a latent space vector and passing it to the

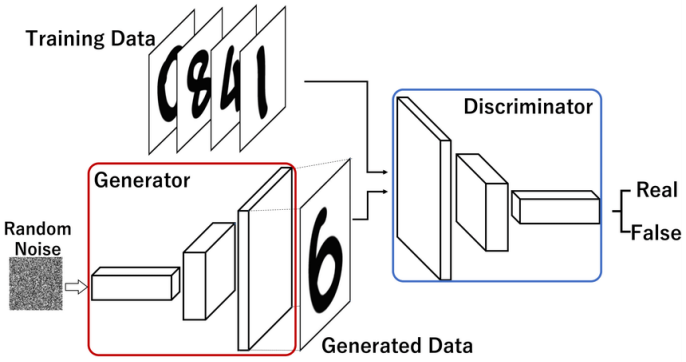


FIG. 2. Typical GAN architecture retrieved from [38].

generator, which has learned to translate it into a realistic image. Fig.2 shows an overview of the original architecture of GAN for generating 2D data, but all the forthcoming developments still hold for 1D data. This early approach has been shown to work well under some hyperparameter configurations [39]. However, early GAN architecture [16] suffers from the significant problems of vanishing gradients and meaningless loss function [40]. Wasserstein GANs [41] (WGAN) were developed to address these issues by making use of the Earth's mover distance estimator, or Wasserstein-1 distance (W_1) [42], which computes the similarities between two distributions. W_1 is evaluated through the discriminator as the training progresses and increases monotonically while never saturating, providing a meaningful loss metric even for two disjoint distributions. Since W_1 is continuous and differentiable, it yields reliable gradients, allowing us to train the discriminator till optimality to obtain high-quality generations. This change of paradigm led Arjovsky et al.[41] to reformulate the optimization problem as :

$$\theta_{opt} = arg\,\min_{\theta} W_1(P_x \| P_{\tilde{x}})\,, \tag{1}$$

where W_1 is evaluated between the real distribution P_x and generated distribution $P_{\tilde{x}}$. Eq.1 can be written as,

$$\theta_{opt} = arg\,\min_{\theta} \max_{\phi:\|D(x,\phi)\|_L \leq 1} L(\phi,\theta) \tag{2}$$

with the discriminator loss:

$$L(\phi,\theta) = -E_{x\sim P_x}\big[D(x,\phi)\big] + E_{\tilde{x}\sim P_{\tilde{x}}}\big[D(\tilde{x},\phi)\big] \tag{3}$$

where D and G refer to the discriminator and the generator with parameters ϕ and θ, respectively. $E_{x\sim P_x}$ indicates that the expression has been averaged over a batch of real samples x, while $E_{\tilde{x}\sim P_{\tilde{x}}}$ has been averaged over a batch of generated samples $\tilde{x}$. The new condition over ϕ in expression Eq.2 imposes a constraint on the discriminator D, which must be 1-Lipschitz continuous [41].

In practice, this can be achieved in two ways: clipping the weights of the discriminator beyond a specific value c[41], or adding a regularization term to the discriminator loss, defined in Eq.3, known as gradient penalty (GP). While the first solution is a poor way to enforce the Lipschitz condition, the second solution has been widely accepted. The mathematical formulation of GP is as follows:

$$L_{tot} = L(\phi,\theta) + \lambda\, GP(\phi) \tag{4}$$

with

$$GP(\phi) = E_{\hat{x}\sim P_{\hat{x}}}\Big[\big(\|\nabla_x D(\hat{x},\phi)\|_2 - 1\big)^2\Big]\,, \tag{5}$$

where λ is known as the regularization parameter, $\|\cdot\|_2$ stands to the L2-norm and $\hat{x}$ is evaluated following:

$$\hat{x} = \tilde{x}\,t + x\,(1-t) \tag{6}$$

with t uniformly sampled $\sim [0,1]$. This method has shown impressive applications such as [43], but it is not restricted to WGANs [44, 45]. Nonetheless, unlike weight clipping, GP cannot enforce the Lipschitz condition everywhere, particularly at the beginning of the training. This can prevent the generator from converging to the optimal solution. To overcome this obstacle, Wei et al. have proposed a second penalization term to add to the loss from Eq.3, called consistency term. They applied their new constraint to two perturbed versions of the real samples x, introducing dropout layers into the discriminator architecture. This ultimately leads to two different estimates noted $D(x')$ and $D(x'')$. The consistency term is defined as follows:

$$\begin{aligned} CT(\phi) = E_{x\sim P_x}\big[max\big(0,\, d(D(x',\phi), D(x'',\phi)) \\ + 0.1\, d(D_{\text{-}}(x',\phi), D_{\text{-}}(x'',\phi)) - M'\big)\big]\,, \end{aligned} \tag{7}$$

where d(.,.) is the L2 metric, $D_{\text{-}}$ stands for the second-to-last layer output of the discriminator, and M' is a constant value. Wei et al. found that controlling the second-to-last layer output helps improve the performance of the WGANs. Thus, the final discriminator loss is then [46]:

$$L_{\text{tot}} = L(\phi,\theta) + \lambda_1\, GP(\phi) + \lambda_2\, CT(\phi)\,, \tag{8}$$

with λ_2 being the consistency parameter. This type of WGAN was called CT-GAN, which is the one that we employ in this work.

B. Network Architecture

The architecture of the networks has been inspired by the work presented in [39] but nearest-neighbour (NN) sampling layers have been preferred over strided convolution layers in the generator structure. The convolution parameters were chosen to be fixed through the generator and discriminator layers with kernel $k = 5$, no padding and stride $s = 1$. $Leaky\ ReLU(\cdot, \alpha = 0.2)$ has been chosen as the activation layer for both discriminator and

generator, with the exception of the output layer of the generator, that uses a $Tanh(\cdot)$ activation, allowing values $\sim [-1, 1]$.

In the generator structure (see Fig.3), we also employ a dilation factor of 2, 4, 6, 8 and 16 for successive layers to enlarge its receptive field and, in turn, its expressivity power, at the exact computational cost [47]. Batch normalization (BN) [48] has been added to the generator architecture to make it both stable and faster to learn. The discriminator structure (see Fig.4) is composed of convolutions on which spectral normalization [49] is employed to stabilize the training. Dropout layers are added, excluding the first and last layers, which is required by the consistency term (Eq.7).

C. Training Data and Procedure

1. *Pre-processing*

The construction of our data set strongly relies on the confidence provided by Gravity Spy. Thus, to create a high confidence data set, we select the blip glitches from L1 (Livingston) and H1 (Hanford) detectors of O2[1] run that have a confidence $c^1_{GS} \geq 0.9$. Glitches are surrounded by stationary and uncorrelated noise, which will hinder the learning of our machine learning method. Therefore, it is necessary to extract glitches from the stream data maintaining their original morphology. For this aim, we employ BayesLine [50] to whiten the glitches locally and BayesWave (BW) [51] to extract the glitches from the uncorrelated noise. BW uses non-orthogonal continuous Morlet-Gabot wavelets to fit and reconstruct the input signal, but the selection of the model is made with a trans-dimensional Reversible Jump Markov Chain Monte Carlo [52] that acquires a trade-off between the complexity of the model and the quality of the fit. The input signal is represented as a set of wavelets whose reconstruction is their addition.

In our particular framework, the input provided to BW is a time series containing the blip glitch that is 2.0 sec long. However, to avoid training the CT-GAN algorithm in irrelevant data and speed up the training phase, the samples of the final training set have 938 data points sampled at 4096 Hz, constituting 0.23 sec of data. Since the reconstruction is not perfect, we lose around 2% and 18% of the data for L1 and H1, respectively (see Table I). To assess the quality of the reconstructions, we inject them in real whitened noise and evaluate it with *Gravity Spy* classifier, selecting blips with a $c^2_{GS} \geq 0.9$ to generate high-quality input data. After this heavy pre-processing, the training data set is composed of around 66% and 50% of the initial data for L1 and H1, respectively.

Moreover, as it was previously mentioned, blips can be found in the frequency band $[30, 250]$ Hz, but BW might introduce certain high-frequency contributions that will hinder the learning of our machine learning algorithm. For illustration, in Fig.5 (left) we plot BW reconstruction (grey), where we coloured the characteristic blip peak (blue) and the high frequency contribution (light blue). To eliminate the high-frequency contribution, we initially set an empirical threshold to remove power excess in the surroundings of the peak. Nonetheless, some high-frequency contributions overlap with the blip and cannot be removed with this method. Thus, to minimize this contribution and generate a smoother input to enhance the learning of our model, we employ regularized Rudin-Osher-Fatemi (rROF) proposed in [53].

TABLE I. Size of the blip set for each detector in the different phases of the pre-processing: selection, reconstruction and evaluation.

Pre-processing	Livingston	Hanford
Num. blips $c^1_{GS} \geq 0.9$	5540	6768
BW output	5461	5612
Num. blips $c^2_{GS} \geq 0.9$	3654	3407
Num. blips $c^2_{GS}, c^3_{GS} \geq 0.9$	3291	2587

This algorithm solves the denoising problem, $s = g+n$, where g is the smooth reconstruction of glitch and n is the noise, as a variational problem. The solution g is computed as follows:

$$g_\lambda = \arg\min_g \{\mathcal{R}(g) + \frac{\lambda}{2}\mathcal{F}(g)\}\,, \tag{9}$$

where $\mathcal{R}(g)$ is the regularization term that constrains the data, which refers to the quality of the smooth reconstruction g. $\mathcal{F}(g)$ is the fidelity term, which measures the L^2-distance between the g glitch and the observed signal s. λ regularises and controls the relative weight of both terms in the equation. It is important to note that his parameter needs to be tunned manually to achieve the desired level of denoising.

To assess the quality of the denoised blip glitches, we use the *Gravity Spy* classifier for different λ parameters again, and we found $\lambda = 0.5$ to be a trade-off between preserving the structure of the glitch and removing the non-smooth high-frequency contribution. In Fig.5 (right), we plot the BW reconstruction denoised with rROF (dashed orange), and the denoised characteristic blip (green). In Fig.5 (bottom), we show the amplitude spectral density (ASD) of the BW reconstruction with and without denoising (grey and dashed orange), as well as the characteristic peak with and without denoising (blue and green) and the original high-frequency contribution (light blue). We can observe that we are able to maintain the structure of the characteristic peak by damping the power of the high-frequency contribution.

[1] Data from GWOSC https://www.gw-openscience.org/data/

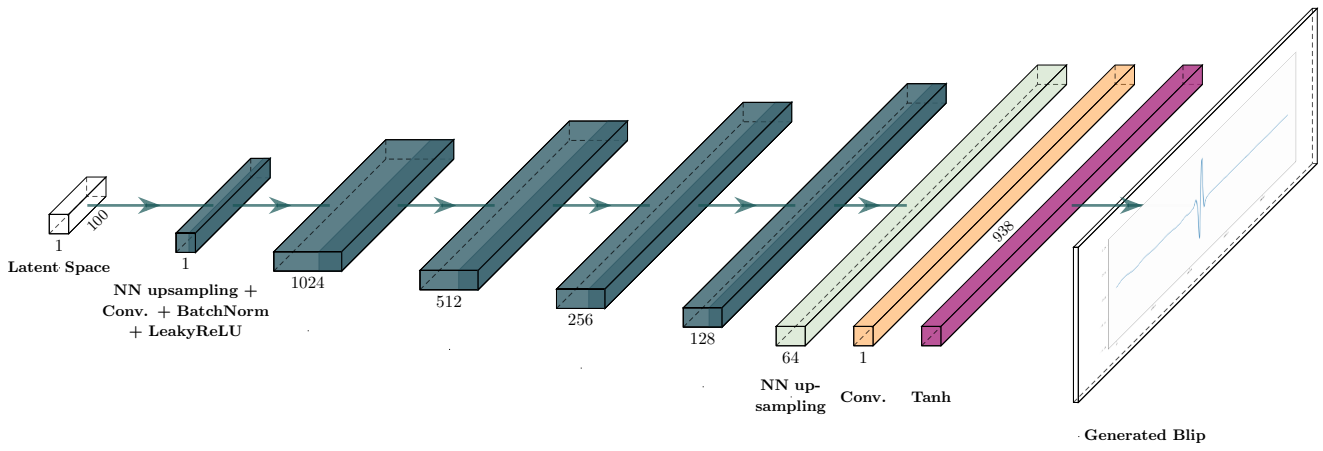


FIG. 3. Generator structure including NN upsampling, convolution layers and LeakyReLU activation.

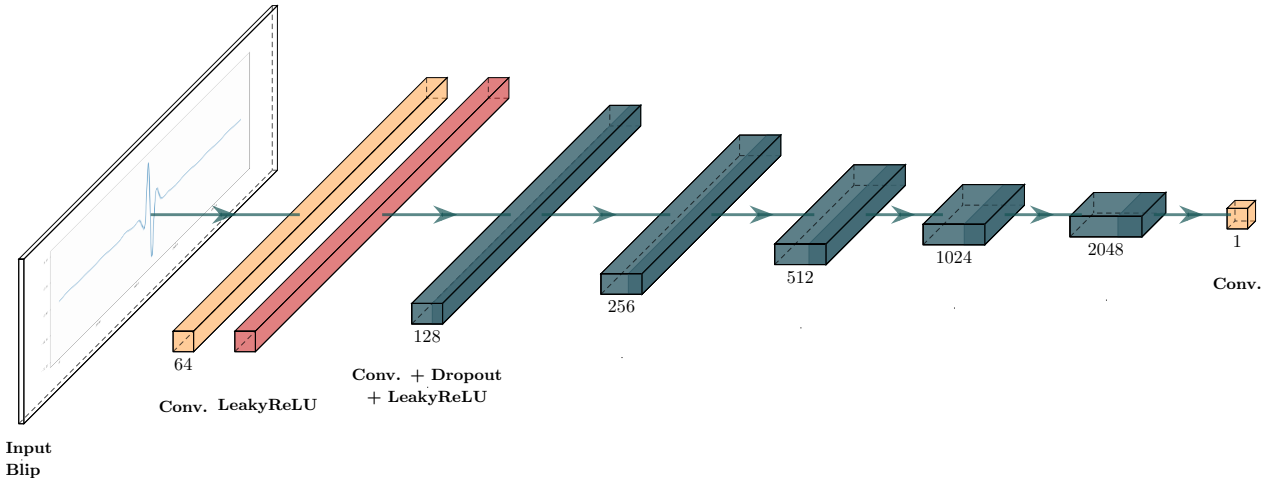


FIG. 4. Discriminator architecture showing strided convolutions, dropout layers and LeakyReLU activations.

To verify that we are able to preserve the structure of blips according to the current state-of-the-art, we compare in Fig.6 the *Gravity Spy* confidence of reconstructed blips c^2_{GS} (blue), against denoised reconstructed blips c^3_{GS} (orange) from L1. As we can observe, both distributions are similar since they have similar means μ_{GS} and standard deviations σ_{GS}. Finally, we select the blip glitches with $c^2_{GS} \geq 0.9$ and $c^3_{GS} \geq 0.9$, to ensure the high quality of the input data of the algorithm.

2. *CT-GAN training procedure*

During the training of the CT-GAN algorithm, both the generator and the discriminator need to be updated at similar rates to acquire stability and guarantee convergence. The task of the discriminator is more difficult since the generated samples that the discriminator intends to classify can be anywhere in the data space and change for each new iteration [54]. Hence, to assure the stability of both networks, we update the discriminator 5 times per update of the generator, for each epoch. We employ RMSProp optimizer [55] with a learning rate $= 10^{-4}$ for both discriminator and generator, and we train the CT-GAN for 500 epochs, where we define an epoch as the number of times the network has passed through the whole dataset. Employing GPU TITAN V with a memory of 96 Gb allowed us to use train our model in ≈ 7.75 h.

To monitor the behaviour of the CT-GAN during the training phase, we represent the generator and the discriminator loss as a function of the epochs in Fig.7. We can observe that both networks stabilize around epoch 100 and continue to oscillate around values close to zero until the training is complete. After several experiments, we concluded that while CT regularised the generator, dropout regularised the discriminator and GP balanced both. This stability can also be observed in the behaviour

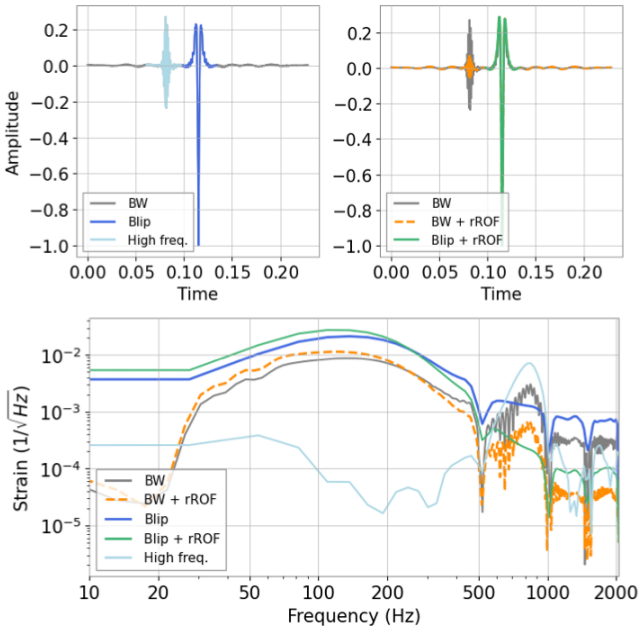


FIG. 5. *(Top left)* Blip glitch reconstructed with BW (grey), where we colour the characteristic blip peak (blue) and the undesired high frequency contribution (light blue). *(Top right)* Blip glitch reconstructed with BW (grey) and denoised with $\lambda = 0.5$ (dashed orange). We colour in green the denoised characteristic blip peak. *(Bottom)* Resulting amplitude spectral density (ASD) for the reconstructed blip with BW (grey) and its denoised version with $\lambda = 0.5$ (dashed orange). We also show the ASD of the characteristic peak with (blue) and without denoising (green), as well as the high frequency contribution (light blue).

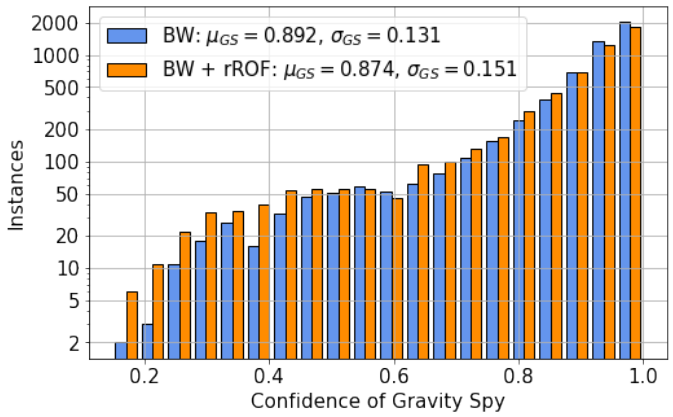


FIG. 6. Comparison between the reconstructed and the denoised population of blip glitches for L1. For the reconstructed set $c^2_{GS} = 0.892 \pm 0.003$ and for the denoised set $c^3_{GS} = 0.874 \pm 0.004$ at 95% confidence level.

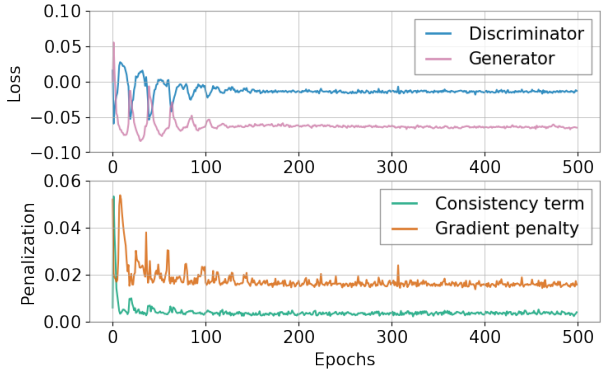


FIG. 7. Graph representing the discriminator loss (blue), generator loss (pink), CT (green) and GP (orange) penalisation as a function of the epochs.

of the CT and GP penalizations in Fig.7, where both terms tend to zero as the network stabilizes. The values that helped the CT-GAN to converge were CT = 5 and GP = 5, with a dropout rate of 0.6. These values were obtained after several experiments, but in future works it would be interesting to employ Optuna [56], which is a hyperparameter optimization framework used to automate hyperparameter searches.

IV. RESULTS

A. Blip Generation

After the training of the CT-GAN, and given a 100-dimensional vector drawn from a normally distributed latent space (as it common in other GAN related works), we are able to generate 10^3 blips a blip from the input distribution of H1 and L1 in ≈ 5 sec for both interefer-ometers. It is relevant to note that each blip has a length $\approx 0.23s$ with an amplitude $\in [-1, 1]$, whitened and sampled at 4096 Hz. In Fig.8 we represent the peak frequency (top panel) and the duration (bottom panel) for the fake population from L1, and we compare it with Tomte and Blips. As an example, we present in Fig.9 different artificial blips from L1 in the time domain, and for visualization, we also compute their Q-transform as in [20]. In the time-frequency representation, we can see that CT-GAN has been able to capture the distinct symmetric 'teardrop' of blips in the expected frequency range $[30, 250]$ Hz. In Fig.8, we compare the peak frequencies of real Tomte and Blip glitches from L1 against our artificial population, where we can observe that the bulk of the distribution of fake blips is aligned with the real blip population. Furthermore, we can observe that in the time representation, we are able to reproduce different morphologies of the characteristic central peak. Even if by visual inspection it would seem that the artificial generations are closely related to the real blips from O2, it is necessary to perform a statistical test to assess the performance of CT-GAN.

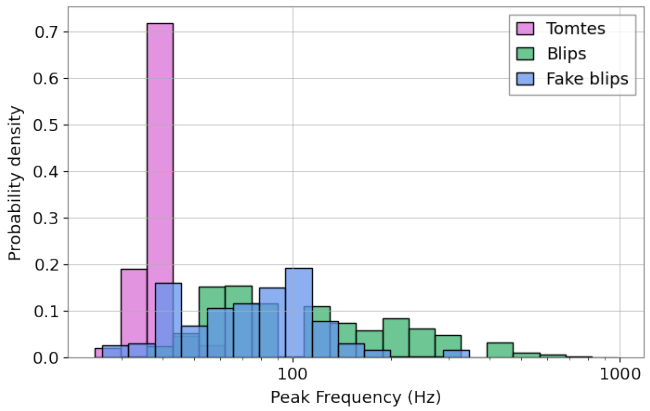


FIG. 8. *(Top)* Peak frequency for Tomte (pink) and Blip (green) from L1 retrieved from Gravity Spy [20], measured with Omicron spectrograms [57]. In blue we plot the peak frequency of the artificial blips from L1.

B. Assessing Performance

We employ four different methods to assess the quality of the population. On the one hand, we employ their Q-scan representation to evaluate our artificial population with the current state-of-the-art. On the other hand, we analyze their morphology in the time domain to take into account the phase information:

- **Gravity Spy classifier:** In order to assess performance using an independent ML classifier, we can inject the generated glitches in real whitened noise from O2 (see Fig.9) and evaluate them with *Gravity Spy*, which will return a confidence value c_{GS} and a class label. We use the same noise strain for each generated glitch to provide the classifier with a fair comparison. Since the generated blip has an amplitude $\in [-1, 1]$, we can re-scale it according to a desired optimal signal-to-noise ratio (ρ_{opt}). For this aim, we relate ρ_{opt} to the scaling parameter α by modifying Eq 4.3 from [58] as:

$$\rho_{opt} = 4\alpha \int_{f_{min}}^{f_{max}} \frac{|\tilde{g}(f)|^2}{S_n(f)} df \,, \tag{10}$$

  where $\tilde{g}(f)$ represents the artificial blip and S_n is the power spectral density (PSD) of the fixed real whitened noise. One of the main drawbacks of this method is that it is computationally intensive (≈ 90 s/glitch) because it is necessary to calculate the Q-transform of the input time series.

- **Wasserstein distance** (W_1): As explained in subsection *ii*, the Wasserstein distance is continuous and never saturating, allowing us to keep track of the quality of the generated samples during the training. For further mathematical details, a formal definition can be found in [41]. This metric is then an adequate tool to compare real and generated glitches. This method is fast and efficient since the computation is performed in the time domain (≈ 0.0026 s/glitch).

- **Match function** (M_f)**:** To compute the similarity between two signals, we can also use the match function, which returns the match between both signals [59]. The match can be defined as the inner product between two normalized signals maximized over time (t) and phase (ϕ) [60],

$$M_f(a, b) := \max_{t,\phi} \langle \hat{a}, \hat{b} \rangle. \tag{11}$$

  Since the signals are noise-free, we do not employ any PSD for normalization. This calculation is performed in the frequency domain, and it is also fast and efficient (≈ 0.0032 s/glitch).

- **Normalized cross-covariance** ($k(X, Y)$)**:** Assuming two random processes X and Y, their cross-covariance between time t_1 and t_2 is defined as:

$$K_{X,Y}(t_1, t_2) \equiv E[(X_{t_1} - \mu(X_{t_1}))\overline{(Y_{t_1} - \mu(Y_{t_1}))}] \tag{12}$$

  To obtain the normalized cross-covariance coefficient, we divide the cross-covariance over the standard deviation of each random process. The maximum value of this magnitude is the metric employed to measure the similarity between two signals, as defined below:

$$k = \max \left(\frac{K_{X,Y}(t_1, t_2)}{\sigma_X \sigma_Y} \right) \tag{13}$$

  This calculation, which is also in time domain, is most efficient (≈ 0.0011 sec/glitch).

1. *Gravity Spy*

For this procedure, we inject each generated blip in real whitened detector noise and re-scale it according to Eq.10 to fix ρ_{opt}. We can compute the confidence of *Gravity Spy* as a function of the optimal SNR $\rho_{opt} \in [0.1, 18.2]$. This process is conducted on 10^3 blip glitches of each detector population.

In Fig.10, we plot the classification labels, with maximum classification probability, for different ρ_{opt} of H1 population, while we present the results of L1 in Appendix A. We can observe that the dominant class is *Blip* and that the number of glitches in this class increments by increasing ρ_{opt}, in opposition to other classes. Interestingly, when $\rho_{opt} = 0.947$, meaning that the artificial blip is not visible by eye in the Q-transform, around 500 artificial glitches are labeled as *Blip*.

One could think that this type of behaviour would be expected since CNNs are able to "see" patterns that are invisible to the human eye, but the classifier is able to recognize glitches up to a certain threshold (Omicron SNR

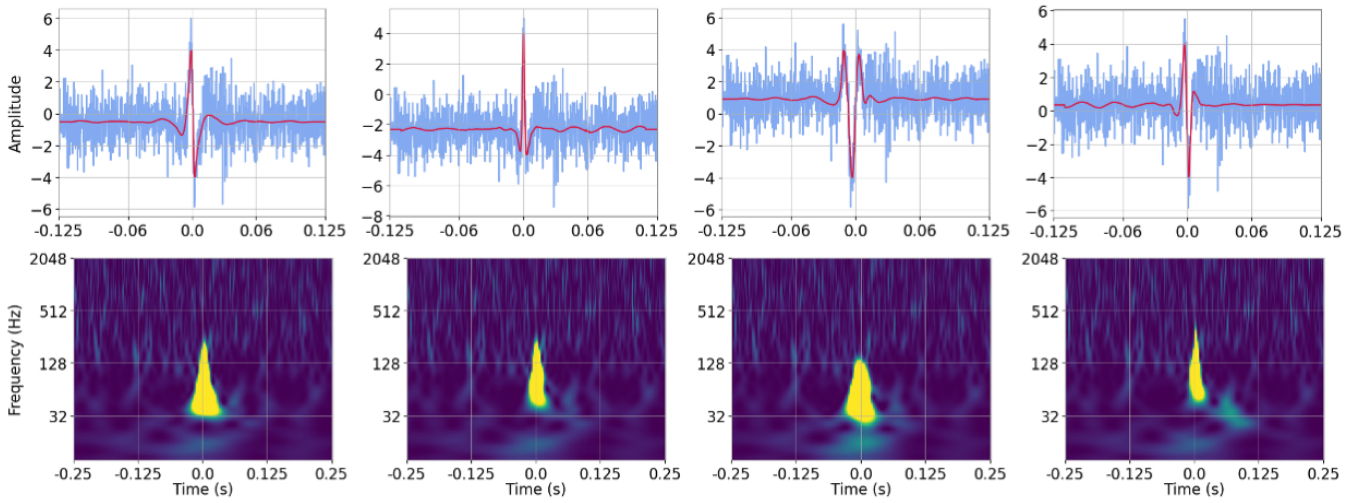


FIG. 9. *(Top row)* Generated blip of L1 plotted as a function of time. In red we represent the rescaled whitened blip and in blue we plot its injection, both in the time domain. *(Bottom row)* We show the Q-transform representation of the generated injected glitches.

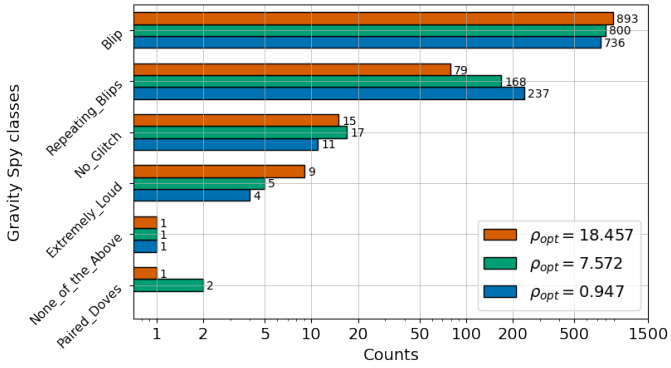


FIG. 10. Histogram of predicted *Gravity Spy* classes for 10^3 generated blips from H1.

≥ 7.5 [57]). Another reason might be that the training set of *Gravity Spy* is imbalanced, so the classifier is biased towards the larger classes such as *Blip*. Hence, it seems that *Gravity Spy* has a certain degree of misclassification, so we employ other metrics to test the performance of our CT-GAN. In the future we will revisit the performance evaluation with independent ML classifiers. For now we focus mainly on similarity metrics as we explain in the following section.

2. *Wasserstein distance, match function and normalised cross-covariance during testing*

To measure the performance of the network, we use some alternative methods, namely Wasserstein distance (W_1), match function (M_f), and normalized cross-covariance (k). These metrics are employed to calculate the similarity between two different artificial blips b_1 and b_2, but we can also use them to calculate the similarity between a single artificial blip b_F and the real population (B_R) or the artificial population (B_F) from each detector. Such procedure is as follows:

1. We use a certain similarity distance m to measure the distance between blip b_j and a population B.

2. For each blip $b_i \in B$ we compute $m_{j,i}(b_j, b_i)$, which yields a set of measurements M_j.

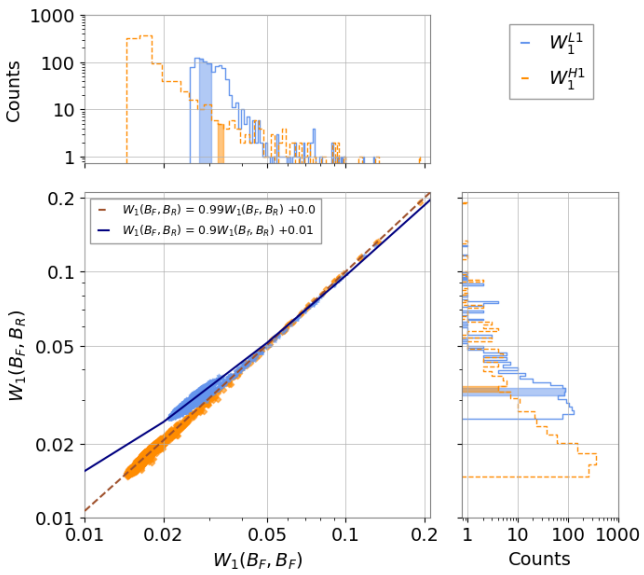
3. We obtain the mean and the standard error of the previous set as $\mu(M_j) \pm \epsilon(M_j)$ at 99.7% confidence interval.

The latter is the measure of similarity between the population B and b_j. Note that the numerical meaning of Wasserstein distance, match function, and normalized cross-covariance are different. For the previous example,

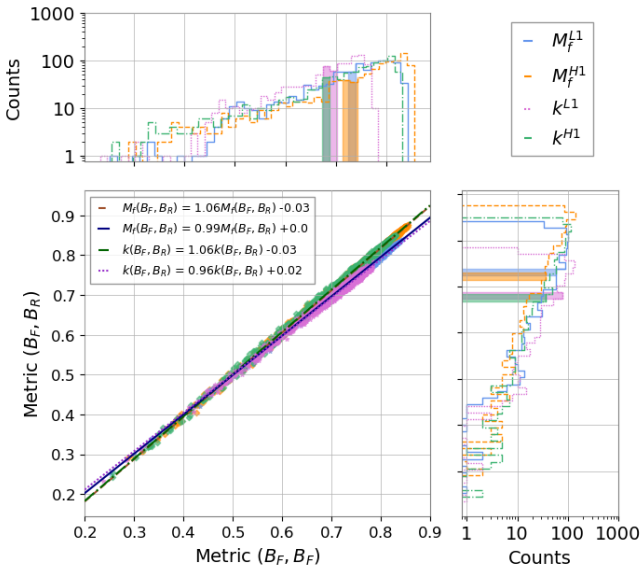
- If b_j is a reliable generation then $W_1(B, b_j) \approx 0$, while $M_f(B, b_j) \approx 1$ and $k(B, b_j) \approx 1$.

- If b_j is an anomalous generation then $W_1(B, b_j) \gg 0$, while $M_f(B, b_j) \ll 1$ and $k(B, b_j) \ll 1$.

Since we are dealing with real data, the real population B_R contains not only blips but also certain misclassifications. If the CT-GAN had learned the underlying distribution of the data, we would expect that the real population B_R and the artificial population B_F had a similar distribution, where reliable generations would be located in the bulk of the distribution. In contrast, anomalous blips would be located in the tails. Hence, under this assumption, we would expect that, given a metric m, the similarity distance between the real and artificial distribution $m(B_R, B_F)$, should be linearly related to the similarity distance of the artificial distribution against



a) Similarity distance: Wasserstein distance (W_1)



b) Similarity distance: match function (M_f) and normalised cross-covariance (k)

FIG. 11. *(Top)* We represent the joint and marginal distributions of $W_1(B_R, B_F)$ and $W_1(B_F, B_F)$ for L1 (blue) and H1 (orange) and their best fit. *(Bottom)* We represent the joint and marginal distributions of the pairs $[M_f(B_R, B_F), M_f(B_F, B_F)]$ and $[k(B_R, B_F), k(B_F, B_F)]$ for L1 (blue and pink) and H1 (orange and green), as well as their best fit. The coloured regions in the marginal distributions represent the confidence interval at 6 standard deviations.

itself, $m(B_R, B_F)$. In Fig.11, we plot the joint and marginal distribution of both comparisons for different similarity distances: Wasserstein distance (W_1), match function (M_f) and normalised cross-covariance (k), and present the results from the least-squares estimate for each detector. Furthermore, in Table I we present the Pearson coefficient resulting from the least-squares estimate, which represents the linear correlation between both variables [61].

TABLE II. Pearson coefficient for different metrics and detectors.

	Livingston	Hanford
Wasserstein distance	0.993	0.999
Match function	0.999	0.999
Normalised cross-covariance	0.996	0.999

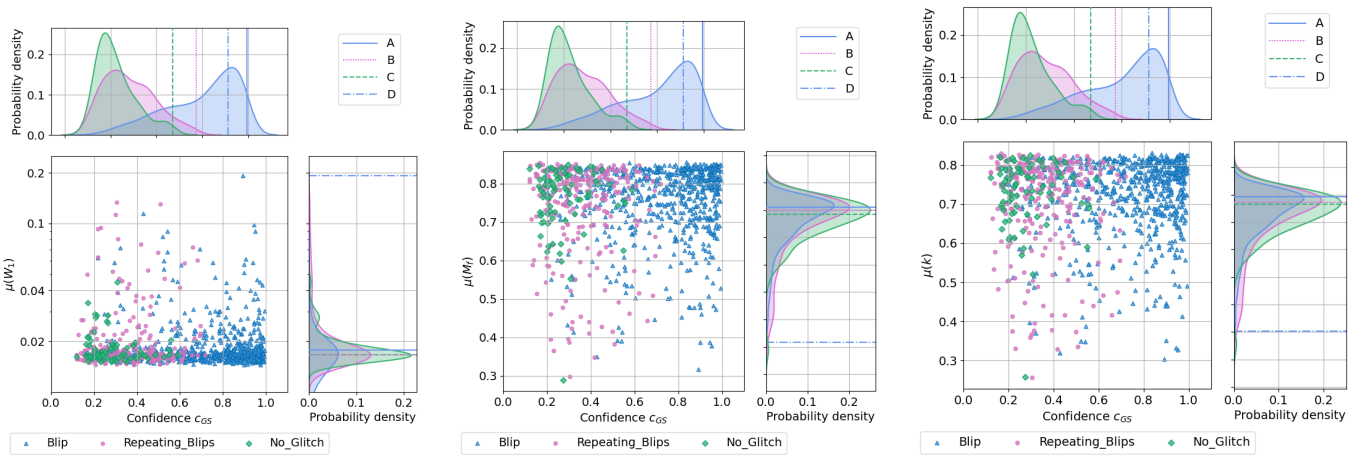
We observe that the resulting slopes (Fig.11) and the Pearson coefficients (Table I) for each metric and each detector are close to 1.0, meaning that both variables have a very strong linear relationship and compatibility. Thus, all similarity distances indicate that the bulk of the population is constituted by reliable blips, with the presence of some anomalous generations that can be identified by fixing an empirical threshold. Therefore, since the generated blips represent the artificial and real populations, we conclude that the CT-GAN has learned the underlying distribution of blips from L1 and H1.

C. Assessing Poor Generations

When dealing with real data, one must bear in mind that certain anomalies might be present in the data. In our particular context, our data sets might contain glitches that have a distinct morphology from the mean of the population. Such differences might not be visible in a Q-transform representation, so *Gravity Spy* might introduce certain miss-classifications that contaminate the input dataset. Since CT-GAN is able to learn the underlying distribution, it can also generate non-blip glitches that are in the tails of the distribution. For certain studies, the presence of anomalies might be counterproductive, so differentiating reliable from anomalous generations is crucial. For this aim, we propose several metrics to identify these miss-generations.

To use Gravity Spy classifier, we inject the generated blips in real whitened noise with a fixed optimal SNR $\rho_{opt} = 18.46$, according to Eq.10. From the classification, we select the generated blips that belong to the three dominant classes: *Blip*, *Repeating_Blips*, and *No_Glitch*.

In Fig.12 we plot the joint and marginal distribution as probability densities of Gravity Spy confidence against the alternative metrics for H1 (see Appendix A for details about L1). We can observe that according to *Gravity Spy Blip*, *Repeating_Blips*, and *No_Glitch* seem to belong to distinct probability densities. However, according to the alternative metrics, the probability densities remain centered according to a certain value for different classes. Furthermore, there seems to be no correlation



a) Wasserstein distance (W_1) b) Match fucntion (M_f) c) Normalised cross-covariance (k)

FIG. 12. Joint and marginal distribution of *Gravity Spy* confidence c_{GS} at $\rho_{opt} = 18.46$ against different metrics for different glitch classes for H1: *Blip*, *Repeating_Blips* and *No_Glitch*. We mark in the marginal distributions selected generated glitches A (solid blue), B (dotted pink), C (dashed green) and D (dash dotted blue).

between Gravity Spy confidence and other metrics in the joint distribution, so to further understand our results, we proceed to inspect the results by selecting examples,

- **Glitch A:** This glitch is labeled as a *Blips* with a high confidence according to *Gravity Spy* ($c_{GS} \approx 0.99$). Furthermore, the chosen metric has situated this glitch in the bulk of the distribution, meaning that it is a reliable blip generation.

- **Glitch B:** This glitch is labeled as a *Repeating_Blips* with a confidence $c_{GS} \approx 0.72$. However, according to our metrics, it is a reliable generation.

- **Glitch C:** This glitch is labeled as a *No_Glitch* with a confidence $c_{GS} \approx 0.59$. However, according to our metrics, it is also a reliable generation.

- **Glitch D:** This glitch is labeled as a *Blips* with a high confidence according to *Gravity Spy* ($c_{GS} \approx 0.89$). Nonetheless, the chosen metric has situated this glitch in the tail of the distribution, meaning that it is an anomalous blip generation.

In Fig.12, we can observe that according to the alternative metrics, glitches A, B and C are situated around the center of the probability density, while glitch D is located in the tails. Moreover, for further visualization in Fig.13, we present the selected in the time domain, and we also plot their Q-transforms. We can observe that while glitches A, B, and C seem to have a similar shape and magnitude, they differ from anomalous glitch D. Moreover, with these metrics, we are able to identify anomalous generations that deceive Gravity Spy classifier, and their exclusion from the generated data set can be performed by imposing a threshold.

D. Limitations

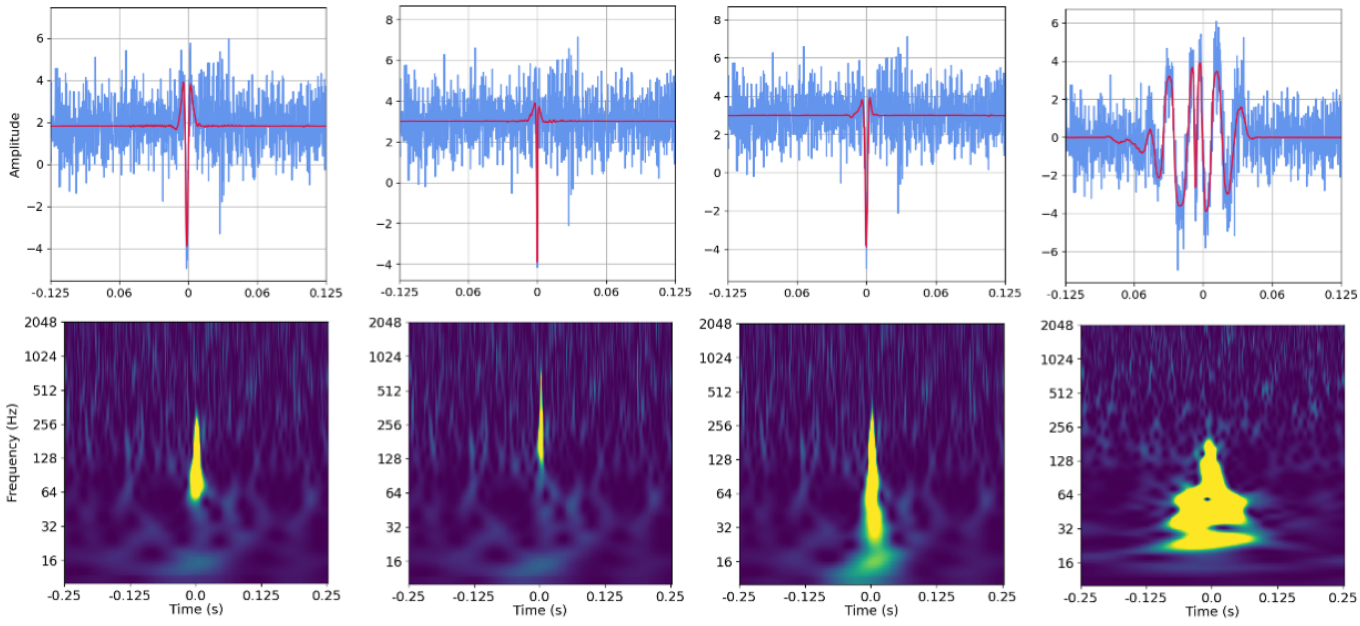
The main shortcoming that we encountered when training the CT-GAN was the limited amount of data preserved after the heavy pre-processing. CT-GAN needs a large amount of samples to learn the underlying distribution, which might be a limitation when extending our methodology to other classes of glitches that are less common in the LIGO/Virgo streams. Nonetheless, some researchers are developing techniques to tackle this limitation that we will explore in future works [62].

Another relevant shortcoming of this study is the fact that the quality of our input data set strongly relies on BW reconstruction and *Gravity Spy* classification. In our particular case, blip glitches have a simple morphology, but some undesired contributions were introduced by BW, and some miss-classifications were introduced by Gravity Spy. Other glitches might be even harder to extract and/or classify with the current state-of-the-art due to their complex form, which in turn will hinder the performance of our CT-GAN. Moreover, longer and more complex glitches will need better architectures to be able to learn the underlying distribution of the data.

V. APPLICATIONS

In the following we provide examples of possible applications that can be explored in future works:

A) Glitch population statistics: Learning the distributions of glitches allows us to understand their populations further and compare their different characteristics. In this way, we can develop statistics to analyze their morphologies, populations, and production rates in more detail as it was discussed in



a) Glitch A b) Glitch B c) Glitch C d) Glitch D

FIG. 13. Time series representation (top row) and Q-scan representation of selected glitches from H1

[63]. For illustration employing generated blips, we have reduced the dimensionality of the artificial population of L1 with Principal Component Analysis (PCA) [64]. By visual inspection, we can see three main clusters that we classify with Gaussian Mixture [2] [66]. Each point represents a single fake blip in PCA space coloured according to their cluster label. Furthermore, we have marked with a star

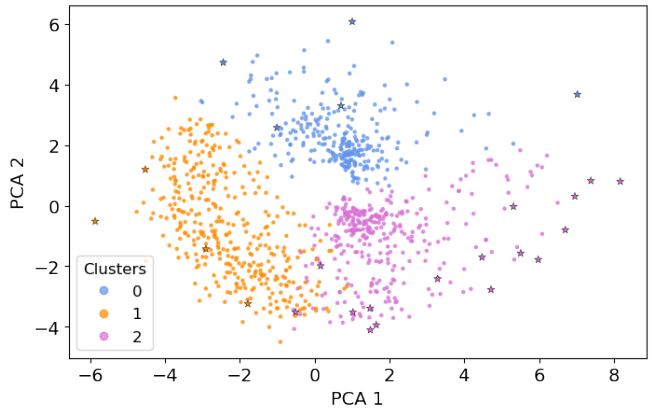


FIG. 14. PCA representation of fake blip population of H1, clustered with Gaussian Mixture. The 5% most anomalous blips according to the distance $W_1(B_F, B_F)$ are marked with a star.

5% of the most anomalous blips present in the population, according to their distance $W_1(B_F, B_F)$. It would be interesting to investigate the differences between the clusters in these distributions in future work. Another possibility would be to link the features of the blip glitches with their representation in the latent space of the CT-GAN, as it was proposed in [67].

B) Glitch template banks: It is well-known that blip glitches have a similar morphology to intermediate black holes (IMBH), which hinders the detection of such events. With our generator, we could create glitch templates to use matched-filtering techniques in unknown signals to compute a ranking statistic and weight it in the likelihood function of detection pipelines. In this way, we would provide another metric to distinguish blip glitches from IMBH. We could use the standard matched-filter method [68] (See Eq.14) to compute the SNR time-series for a specific glitch template. However, performing a matched filtering operation for a large glitch bank will be a huge task as computational time will increase drastically. We need to handle the scalability issue of the computational time of performing matched filtering with the increased number of glitch templates as we would expect to manage many glitch templates. We can resolve this scalability issue if we adapt the matched filtering framework used in the GstLAL [69, 70] pipeline for the searches of GW signals from CBC sources. We observed that a few numbers of basis obtained using Singular Value Decomposition (SVD) [71–74] can also represent the glitch templates, and those basis

[2] For both algorithms, we employ Scikit-learn implementation [65]

can be used to get the matched filter output quickly. The computational time-complexity of matched filtering can be reduced as the required number of basis vectors is much less than the number of glitch templates. To show the efficacy of this framework, we generated 10^3 glitches for the L1 detector using our proposed CT-GAN-based glitch generator. We used 1 second data, sampled at 4096 Hz for this study. The data contains an injected glitch and colored Gaussian noise with aLIGO Zero Detuned High Power (ZDHP) noise power spectral density [75]. Since the generated glitches are around 0.23 second (938 data points) sampled at 4096 Hz, we padded them with zero and made them 1 sec long to generate the noisy data. The amplitudes of the injected glitch were adjusted for a target Signal to Noise Ratio (SNR) of 10. Further, we used ZDHP to whiten the data and the glitch templates. We computed the SNR time-series for each glitch template based on (a) standard matched-filter method [68] as follows:

$$\langle s(t), g(t)\rangle = 4\,\mathrm{Re}\int_0^\infty \frac{\tilde{s}(f)\,\tilde{g}^*(f)}{S_n(f)}\,df\,, \qquad (14)$$

where the term $S_n(f)$ is defined as the one-sided power spectral density (PSD). The square root of Eq.14 is termed as SNR.

(b) SVD based matched filter [72] in which a set of few top basis vectors have been computed from glitch-matrix first. Since each glitch template has 4096 data points, therefore the dimension of the glitch matrix is of size $10^3 \times 4096$ after stacking all the glitches together. After that, the basis vectors are matched filter against data, and the SNR time-series has been computed by combining coefficients of each glitch and matched filter output obtained based on basis and data. For our example, we obtained that 10 top-basis vectors are sufficient to represent those 10^3 glitches, as it can be observed in Fig. 15. It shows that the singular values of a set of 10^3 glitches are fall steeply, which implies a few top-basis (e.g., $10, 20$) can be used to represent those glitches. We have chosen the number of top-basis $(\ell) = 1, 5, 10$ and reconstructed the glitches in our analysis. We have computed the reconstruction error for each glitch as follows:

$$\epsilon_\alpha = \frac{\|g_\alpha - \hat{g}_\alpha\|_2}{\|g_\alpha\|_2}\,; \alpha = 1, 2, \cdots, 10^3 \qquad (15)$$

where $\hat{g}_\alpha$ is the reconstructed whitened glitch based on $\ell = 1, 5, 10$ basis vectors respectively and $\|\|_2$ represents L_2 norm, and α is the number of total glitch templates. We also computed the fractional SNR-loss [72] for each glitch templates based on following definition:

$$\frac{\delta\rho_\alpha}{\rho_\alpha} = \frac{|\rho_\alpha| - |\hat{\rho}_\alpha|}{|\rho_\alpha|}\,; \alpha = 1, 2, \cdots, 10^3 \qquad (16)$$

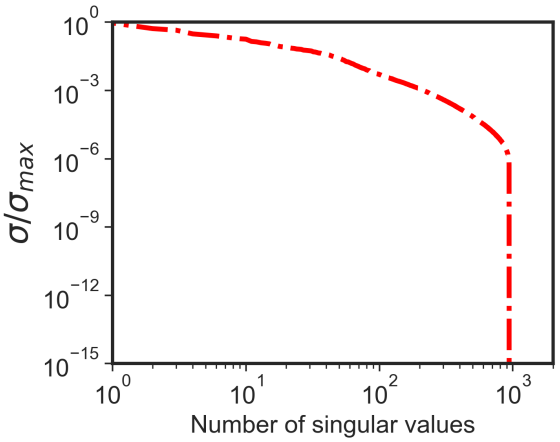


FIG. 15. The singular values (σ) are obtained from a set of 10^3 whitened glitches using SVD [71], normalized by the maximum singular values ($\sigma_{\max}$). The glitches are generated from CT-GAN. The spectrum of singular values is seen to fall sharply, implying only a few singular values (e.g., $\ell = 10$), and corresponding basis vectors are sufficient to represent the glitches. See the Fig.16 in which the relative reconstruction error for these glitches has been shown based on $\ell = 1, 5, 10$. For performing SVD based matched filtering for glitch templates, we followed the framework presented in [72].

With the increasing number of basis, the relative reconstruction error should be decreased. To establish this statement, in Fig.16, we choose three different cases with varying $\ell = 1, 5, 10$. Fig.16 shows the probability density of the relative error ϵ_α for $\ell = 1, 5, 10$ respectively. The figure shows that relative error is less for $\ell = 10$, whereas the relative error is high for $\ell = 1$. Similarly, we obtained the fraction SNR loss for all glitch templates for these three cases. Fig. 17 shows the construction of glitch and SNR time-series based on $\ell = 1, 5, 10$ number of basis respectively. Both plots show that $\ell = 10$ is sufficient to reconstruct the whitened glitches and represent the SNR time series. If we increase the number of basis, the reconstruction errors ($\frac{\delta\rho_\alpha}{\rho_\alpha}$, ϵ_α) can be reduced but matched filtering cost would increase. Hence, we need to choose a minimal set of the basis for which computation cost and also the reconstruction errors are low. We have chosen $l = 10$ as that minimal number for this specific example.

In a follow-up work, we will explore the possibility the construction of a glitch bank construction, with a discussion on how to obtain ranking statistics, and signal consistency tests.

C) Mock data challenges: With our methodology, we are able to generate glitches in the time domain. The user could generate as many glitches as necessary, selecting the ones that represent best the real distribution and injecting them in real detector

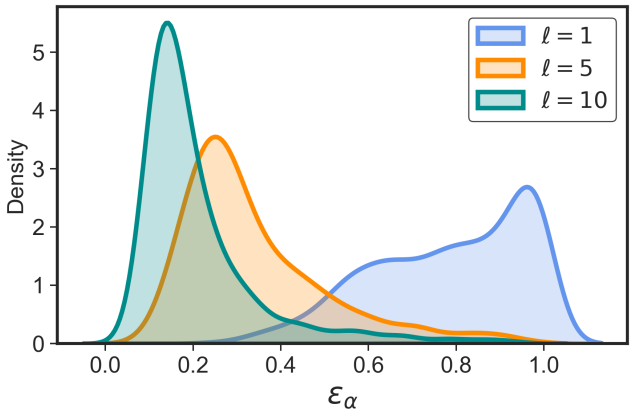


FIG. 16. The plot shows the distribution of relative errors for the reconstruction of the 10^3 whitened glitches generated using CT-GAN. The relative error (ϵ_α) is calculated for each case.

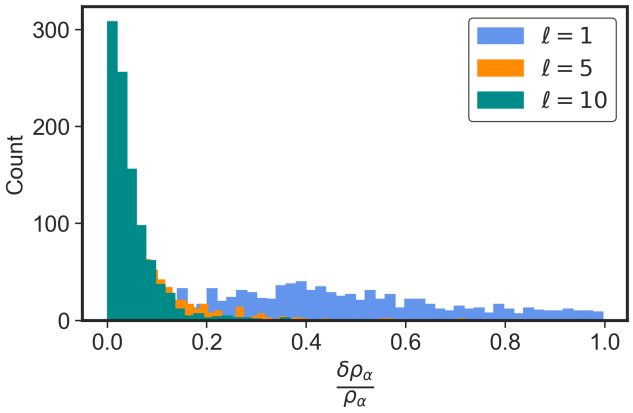


FIG. 17. This figure shows the histogram based on the fractional SNR loss ($\frac{\delta\rho_\alpha}{\rho_\alpha}$) for a set of glitches (10^3). For each glitch template, the SNR time-series were obtained based on (a) Standard matched-filter scheme and (b) SVD based matched filtering framework presented in [72] by varying the top-basis numbers as $\ell = 1, 5, 10$ respectively.

noise to create a realistic data challenge. Moreover, since certain anomalies are generated, those can also be selected to stress-test analysis algorithms. As a preliminary test, we inject some blip glitches in the O3a data to evaluate how they will impact the long-duration analysis with a dedicated neural network called ALBUS [35]. For visualization, we present the output in the right panel of Fig.18. Since a time resolution is much larger than the glitch duration (i.e., < 0.3 s), the injected glitch appears as a vertical line. The structure of the glitch is fully recovered and allows to reveal the detection capability of ALBUS. As suggested in [37], when learning different classes of glitches, we could also interpolate between them to generate hybrid classes. This hybrid dataset could be employed to discover unknown classes of glitches and improve the efficiency of detection algorithms.

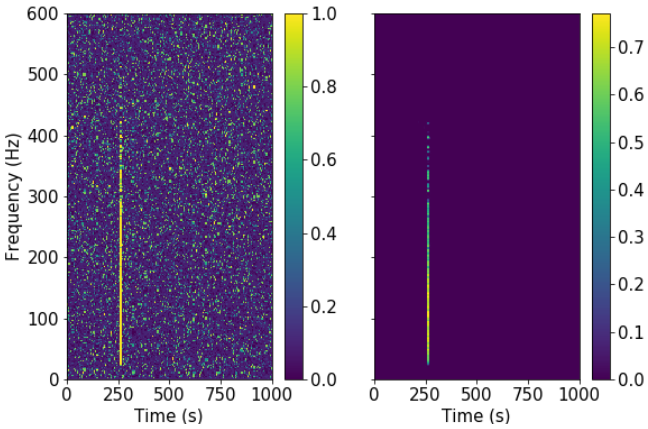


FIG. 18. Example of glitch injection. The left image shows the input time-frequency map while the right panel shows the output of ALBUS.

D) New glitch detection: Once the network has learned the underlying distribution of the data, with certain modifications, it can output how likely it is for an unknown signal to belong to the known distribution. This metric can detect anomalous generations and provide feedback to classification algorithms. For example, with this information *Gravity Spy* could re-classify certain anomalies, which could imply the definition of new glitch classes and their further characterization.

E) Improving glitch classification: One of the main challenges of working with real data is to deal with imbalanced data sets. With our methodology, once more classes are learned, we could generate balanced data sets to improve the accuracy of classification algorithms.

VI. CONCLUSION

In this work, we have developed a methodology to generate artificial blip glitches from real data using a ML algorithm known as GAN. To be able to generate these glitches, the input blips need to be processed: the signals are selected from Gravity Spy data to be reconstructed with BayesWave and smoothed with the rROF algorithm. Because of this heavy processing, only around 66% and 50% of the initial data from L1 and H1 is preserved.

Due to the instability of GAN algorithms, in this particular research, we trained a CT-GAN [46]. The network uses Wasserstein distance as a loss function, which allows it to train its discriminator to optimality. The network is penalized heavily to avoid training instabilities and to learn the underlying distribution of blips accurately.

To assess the performance of CT-GAN, we generate a population of 10^3 blip glitches for both H1 and L1. The quality measurements employed are *Gravity Spy* classifier and similarity distances, namely, Wasserstein distance (W_1), match function (M_f) and normalized cross-covariance (k). The results of these metrics indicate that the neural network was able to learn the underlying distribution of blip glitches from H1 and L1, despite the presence of some anomalous generations due to imperfections of the input data set. Furthermore, it has been observed that the similarity distances are able to detect miss-classifications from glitch classifiers.

In this proof-of-concept investigation, we have demonstrated that it is possible to isolate blip glitches from their surrounding noise and learn their underlying distribution with an ML-based method in the time domain, providing several examples of its usage. This methodology allows us to generate better quality data, and it provides us with flexibility that would be challenging to achieve with Q-transforms. The long-term goal of this investigation is to learn other classes of glitches and create an open-source interface for producing real data in the time domain.

ACKNOWLEDGEMENT

The authors thank Chris Messenger, Siddharth Soni, Jess McIver, Marco Cavaglia, Alejandro Torres-Forné and Harsh Narola for their useful comments. V.B. is supported by the Gravitational Wave Science (GWAS) grant funded by the French Community of Belgium, and M.L., S.C and A.R are supported by the research program of the Netherlands Organisation for Scientific Research (NWO). The authors are grateful for computational resources provided by the LIGO Laboratory and supported by the National Science Foundation Grants No. PHY-0757058 and No. PHY-0823459. This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation.

[1] B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), "Observation of gravitational waves from a binary black hole merger," Phys. Rev. Lett. **116**, 061102 (2016).
[2] J. Aasi et al. (LIGO Scientific), "Advanced LIGO," Class. Quant. Grav. **32**, 074001 (2015), arXiv:1411.4547 [gr-qc].
[3] F. Acernese et al. (VIRGO), "Advanced Virgo: a second-generation interferometric gravitational wave detector," Class. Quant. Grav. **32**, 024001 (2015), arXiv:1408.3978 [gr-qc].
[4] B. P. Abbott et al. LIGO Scientific Collaboration and Virgo Collaboration, "GW170814: A Three-Detector Observation of Gravitational Waves from a Binary Black Hole Coalescence," Phys. Rev. Lett. **119**, 141101 (2017), arXiv:1709.09660 [gr-qc].
[5] B. P. Abbott et al., "GW190425: Observation of a compact binary coalescence with total mass ~ 3.4 m $\odot$," The Astrophysical Journal **892**, L3 (2020).
[6] B. P. Abbott et al. (LIGO Scientific, Virgo), "GWTC-1: A Gravitational-Wave Transient Catalog of Compact Binary Mergers Observed by LIGO and Virgo during the First and Second Observing Runs," Phys. Rev. X **9**, 031040 (2019), arXiv:1811.12907 [astro-ph.HE].
[7] R. Abbott et al. (LIGO Scientific, Virgo), "GWTC-2: Compact Binary Coalescences Observed by LIGO and Virgo During the First Half of the Third Observing Run," Phys. Rev. X **11**, 021053 (2021), arXiv:2010.14527 [gr-qc].
[8] R. Abbott et al. (LIGO Scientific, VIRGO, KAGRA), "GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo During the Second Part of the Third Observing Run," (2021), arXiv:2111.03606 [gr-qc].
[9] D. Reitze et al., "Cosmic Explorer: The U.S. Contribution to Gravitational-Wave Astronomy beyond LIGO," Bull. Am. Astron. Soc. **51**, 035 (2019), arXiv:1907.04833 [astro-ph.IM].
[10] P. Amaro-Seoane et al. (LISA), "Laser Interferometer Space Antenna," (2017), arXiv:1702.00786 [astro-ph.IM].
[11] M. Punturo et al., "The Einstein Telescope: A third-generation gravitational wave observatory," Class. Quant. Grav. **27**, 194002 (2010).
[12] S. Soni et al., "Reducing scattered light in ligo's third observing run," arXiv: Instrumentation and Methods for Astrophysics (2020).
[13] M. Cabero et al., "Blip glitches in Advanced LIGO data," Class. Quant. Grav. **36**, 15 (2019), arXiv:1901.05093 [physics.ins-det].
[14] D. Davis et at. (LIGO), "LIGO detector characterization in the second and third observing runs," Class. Quant. Grav. **38**, 135014 (2021), arXiv:2101.11673 [astro-ph.IM].
[15] F. Acernese et al. (Virgo), "Virgo Detector Characterization and Data Quality during the O3 run," (2022), arXiv:2205.01555 [gr-qc].
[16] I. J. Goodfellow et al., "Generative adversarial nets," (2014).
[17] J. Powell et al., "Classification methods for noise transients in advanced gravitational-wave detectors," Classical and Quantum Gravity **32**, 215012 (2015).
[18] R. Obaid S. Mukherjee and B. Matkarimov, "Classification of glitch waveforms in gravitational wave detector characterization," Journal of Physics: Conference Series **243**, 012006 (2010).
[19] J. Powell et al., "Classification methods for noise transients in advanced gravitational-wave detectors II: performance tests on Advanced LIGO data," Classical and Quantum Gravity **34**, 034002 (2017).
[20] M. Zevin et al., "Gravity spy: integrating advanced ligo detector characterization, machine learning, and citizen science." Classical and quantum gravity **34 No 6** (2017).
[21] B. P. Abbott et al., "Effects of data quality vetoes on a search for compact binary coalescences in advanced

LIGO's first observing run," Classical and Quantum Gravity **35**, 065010 (2018).
[22] B. P. et al. Abbott (LIGO Scientific, Virgo), "Characterization of transient noise in Advanced LIGO relevant to gravitational wave signal GW150914," Class. Quant. Grav. **33**, 134001 (2016), arXiv:1602.03844 [gr-qc].
[23] Y. Bengio I. Goodfellow and A. Courville, Deep Learning (MIT Press, 2016) http://www.deeplearningbook.org.
[24] D. George and E. A. Huerta, "Deep neural networks to enable real-time multimessenger astrophysics," Phys. Rev. D **97**, 044039 (2018).
[25] H. Gabbard et al., "Matching matched filtering with deep networks for gravitational-wave astronomy," Phys. Rev. Lett. **120**, 141103 (2018).
[26] A. Menéndez-Vázquez et al., "Searches for compact binary coalescence events using neural networks in the ligo/virgo second observation period," Physical Review D **103** (2021), 10.1103/physrevd.103.062004.
[27] P. Krastev, "Real-time detection of gravitational waves from binary neutron stars using artificial neural networks," Physics Letters B **803**, 135330 (2020).
[28] G. Baltus et al., "Convolutional neural networks for the detection of the early inspiral of a gravitational-wave signal," Physical Review D **103**, 102003 (2021).
[29] H. Yu et al., "Early warning of coalescing neutron-star and neutron-star-black-hole binaries from the nonstationary noise background using neural networks," Phys. Rev. D **104**, 062004 (2021).
[30] M. López et al., "Deep learning for core-collapse supernova detection," Phys. Rev. D **103**, 063011 (2021).
[31] I. Heng M. Chan and C. Messenger, "Detection and classification of supernova gravitational waves signals: A deep learning approach," Physical Review D **102** (2020).
[32] G. Nurbek S. Mukherjee and O. Valdez, "Study of efficient methods of detection and reconstruction of gravitational waves from nonrotating 3d general relativistic core collapse supernovae explosion using multilayer signal estimation method," Phys. Rev. D **103**, 103008 (2021).
[33] M. Razzano and E. Cuoco, "Image-based deep learning for classification of noise transients in gravitational wave detectors," Classical and Quantum Gravity **35**, 095016 (2018).
[34] E. Cuoco et al., "Enhancing gravitational-wave science with machine learning," Machine Learning: Science and Technology **2**, 011002 (2020).
[35] V. Boudart and M. Fays, "A machine learning algorithm for minute-long Burst searches," (2022), arXiv:2201.08727 [gr-qc].
[36] J. Long E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence **39**, 640–651 (2017).
[37] J. McGinn et al., "Generalised gravitational wave burst generation with generative adversarial networks," Class. Quant. Grav. **38**, 155005 (2021), arXiv:2103.01641 [astro-ph.IM].
[38] S. Harada et al., "Biosignal generation and latent variable analysis with recurrent generative adversarial networks," IEEE Access **7**, 144292–144302 (2019).
[39] L. Metz A. Radford and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," (2016).
[40] L. Weng, "From gan to wgan," ArXiv **abs/1904.08994** (2019).
[41] M. Arjovsky et al., "Wasserstein generative adversarial networks," (2017).
[42] L. V. Kantorovich, "Mathematical methods of organizing and planning production," Management Science **6**, 366–422 (1960).
[43] T. Karras et al., "Progressive growing of GANs for improved quality, stability, and variation," (2018).
[44] T. Salimans et al., "Improved techniques for training gans," (2016).
[45] A. Geiger L. Mescheder and S. Nowozin, "Which training methods for GANs do actually converge?" (2018).
[46] X. Wei et al., "Improving the improved training of wasserstein gans: A consistency term and its dual effect," (2018).
[47] N. Kalchbrenner A. Van Den Oord and K. Kavukcuoglu, "Pixel recurrent neural networks," (2016).
[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," ArXiv **abs/1502.03167** (2015).
[49] I. Gulrajani et al., "Improved training of wasserstein gans," (2017).
[50] T. B Littenberg and N. J. Cornish, "Bayesian inference for spectral estimation of gravitational wave detector noise," Phys. Rev. D **91**, 084034 (2015), arXiv:1410.3852 [gr-qc].
[51] N.J. Cornish and T.B. Littenberg, "Bayeswave: Bayesian inference for gravitational wave bursts and instrument glitches," Classical and Quantum Gravity **32**, 135012 (2015).
[52] J.P. Green, "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," Biometrika **82**, 711–732 (1995), https://academic.oup.com/biomet/article-pdf/82/4/711/699533/82-4-711.pdf.
[53] A. Torres et al., "Total-variation-based methods for gravitational wave denoising," Phys. Rev. D **90**, 084029 (2014).
[54] N. Kodali et al., "How to train your dragan," ArXiv **abs/1705.07215** (2017).
[55] T. Tieleman et al., "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural networks for machine learning **4**, 26–31 (2012).
[56] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama, "Optuna: A next-generation hyperparameter optimization framework," Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019).
[57] F. Robinet et al., "Omicron: a tool to characterize transient noise in gravitational-wave detectors," SoftwareX **12**, 100620 (2020), arXiv:2007.11374 [astro-ph.IM].
[58] B. Allen et al., "Findchirp: An algorithm for detection of gravitational waves from inspiraling compact binaries," Phys. Rev. D **85**, 122006 (2012).
[59] A. Nitz et al., "gwastro/pycbc," (2021).
[60] S. A. Sengupta S. Roy and P. Ajith, "Effectual template banks for upcoming compact binary searches in advanced-ligo and virgo data," Phys. Rev. D **99**, 024048 (2019).
[61] R. E. Walpole et al., Probability and Statistics for Engineers and Scientists (Pearson Education, 2007).

[62] T. Karras et al., "Training generative adversarial networks with limited data," ArXiv **abs/2006.06676** (2020).
[63] G. Ashton et al., "Parameterised population models of transient non-Gaussian noise in the LIGO gravitational-wave detectors," arXiv e-prints , arXiv:2110.02689 (2021), arXiv:2110.02689 [gr-qc].
[64] I. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374**, 20150202 (2016).
[65] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research **12**, 2825–2830 (2011).
[66] D. Reynolds, "Gaussian mixture models," in Encyclopedia of Biometrics, edited by Stan Z. Li and Anil Jain (Springer US, Boston, MA, 2009) pp. 659–663.
[67] Y. Shen et al., "Interpreting the latent space of gans for semantic face editing," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) , 9240–9249 (2020).
[68] B. Allen et al., "Findchirp: An algorithm for detection of gravitational waves from inspiraling compact binaries," Physical Review D **85**, 122006 (2012).
[69] K. Cannon et al., "Gstlal: A software framework for gravitational wave discovery," SoftwareX **14**, 100680 (2021).
[70] C. Hanna et al., "Fast evaluation of multidetector consistency for real-time gravitational wave searches," Physical Review D **101**, 022003 (2020).
[71] C. F. Van Loan and G. Golub, "Matrix computations (johns hopkins studies in mathematical sciences)," Matrix Computations (1996).
[72] K. Cannon et al., "Singular value decomposition applied to compact binary coalescence gravitational-wave signals," Physical Review D **82**, 044025 (2010).
[73] S. Kulkarni et al., "Random projections in gravitational wave searches of compact binaries," Physical Review D **99**, 101503 (2019).
[74] A. Reza et al., "Random projections in gravitational-wave searches from compact binaries ii: efficient reconstruction of the detection statistic," arXiv:2101.03226 (2021).
[75] D. Shoemaker et al., "Advanced ligo anticipated sensitivity curves," LIGO Document LIGO-T0900288-v3 (2010).

Appendix A: Results for blip distribution from L1

This appendix presents the results of blips from the L1 distribution, which are compatible with the H1 population. In Fig.19, we present a histogram of the classes assigned by *Gravity Spy* to a population of 10^3 artificial blips. As in Section IV, we can also observe that the three dominant classes are *Blip*, *Repeating_Blips* and *No_Glitch*, and as we increase the optimal SNR ρ_{opt}, the number of artificial glitches classified as *Blip* increases. As we stated before,

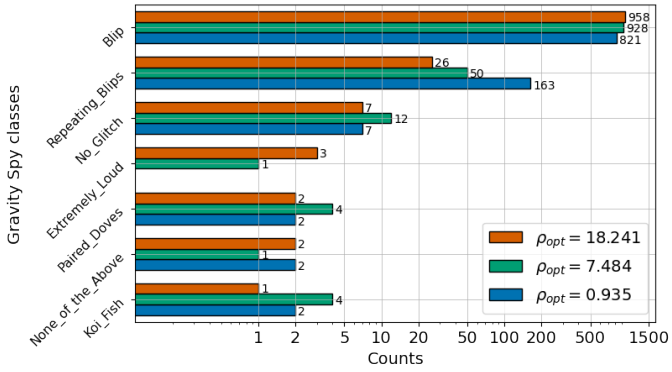
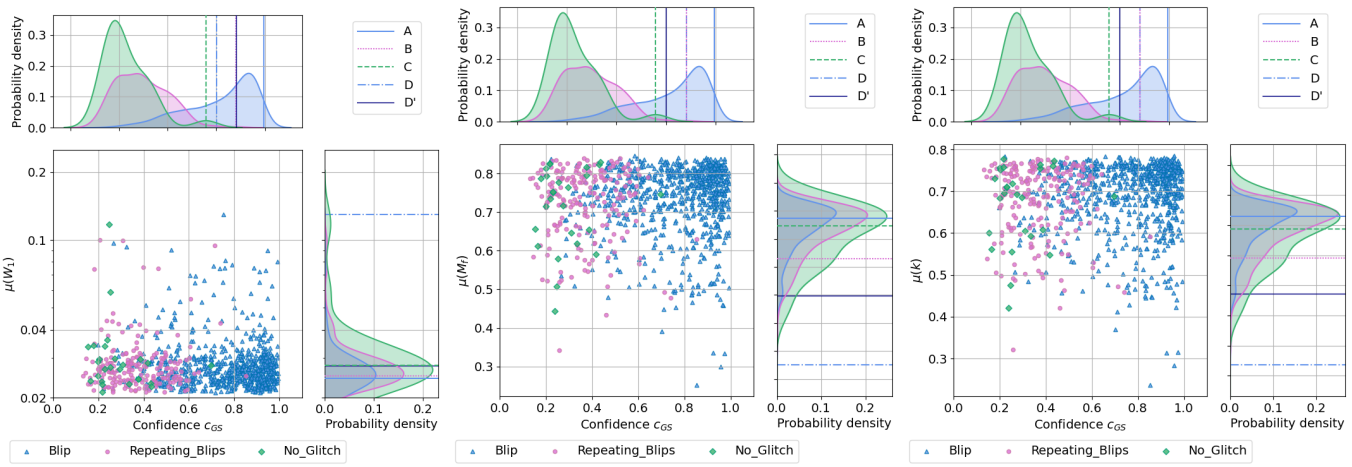


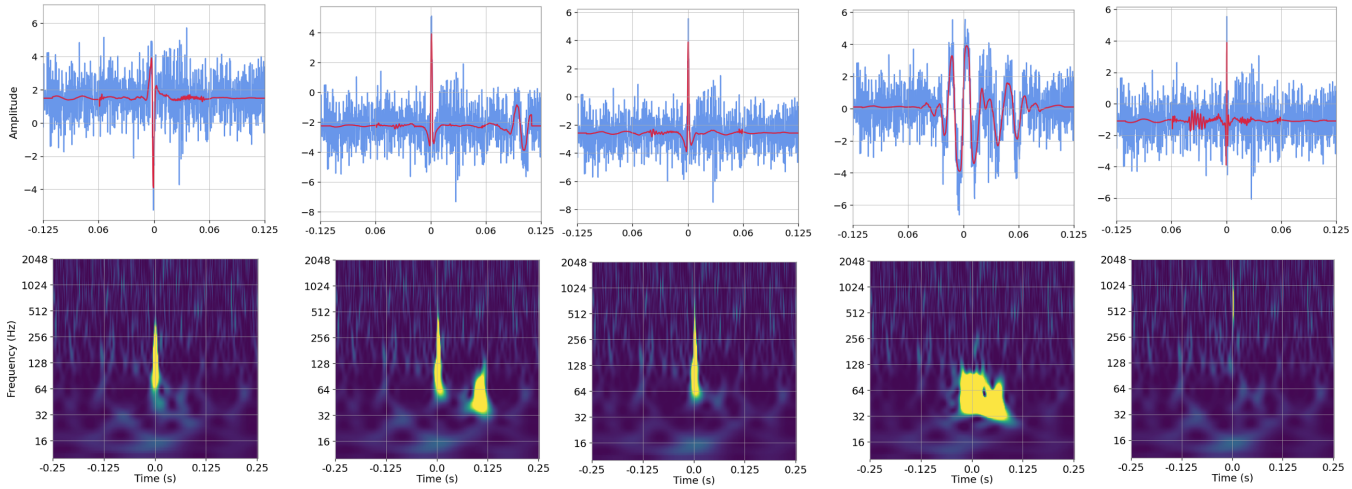
FIG. 19. Histogram of predicted *Gravity Spy* classes for 10^3 blips from L1.

Gravity Spy classifier seems to be biased towards *Blip* class, since at very low ρ_{opt}, the network will be unable to see the glitches. Another interesting question would be to assess the influence of the detector noise in the classification task of Gravity Spy. Similarly to Fig. 12, we present in Fig. 20 the confidence of *Gravity Spy* as a function of alternative metrics for the dominant classes. In Fig.20, we can also observe that there is no apparent correlation between the measurements and the confidence provided by *Gravity Spy* classifier. To inspect the results, we select certain glitches according to the definitions in IV B. Note that the anomalous glitch found by Wasserstein distance (labeled as D) does not coincide with the one found by match function and normalized cross-covariance (labeled as D'). *Gravity Spy* was able to correctly classify with a high confidence glitch A and B, but glitches C, D, and D' are misclassified. For visualization and a better understanding of the results, we plot in Fig.21 the Q-transforms and the time series injected in real whitened noise of the selected glitches. While glitch A is classified by *Gravity Spy* as a perfect glitch, glitch C is miss-classified as *No_Glitch*, although their Q-transforms look similar. It is interesting to mention that the GAN was able to generate a *Repeating_Blip* because some repeating blips are present in the input data set.



a) Wasserstein distance (W_1) b) Match fucntion (M_f) c) Normalised cross-covariance (k)

FIG. 20. Joint and marginal distribution of *Gravity Spy* confidence c_{GS} at $\rho_{opt} = 18.46$ against different metrics for different glitch classes for L1: *Blip*, *Repeating_Blips* and *No_Glitch*. We mark in the marginal distributions selected glitches A (solid blue), B (dotted pink), C (dashed green) and D (dash dotted blue).



a) Glitch A b) Glitch B c) Glitch C d) Glitch D e) Glitch D'

FIG. 21. Time series representation (top row) and Q-scan representation of selected glitches from L1

Glitches D and D', which are misclassified by *Gravity Spy*, are situated in the tail of the distribution of the similarity distances. While glitch D has a shape very different from a standard blip, glitch D' has a very narrow peak.