Machine Learning in Astronomy: Possibilities and Pitfalls
Proceedings IAU Symposium No. 368, 2025
J. McIver, A. Mahabal & C. Fluke, eds.
doi:10.1017/S1743921323001047

Simulating transient burst noise with `gengli`

Melissa Lopez[1,2], Vincent Boudart[3], Stefano Schmidt[1,2] and Sarah Caudill[1,4,5]

[1]Institute for Gravitational and Subatomic Physics (GRASP), Utrecht University, Princetonplein 1, 3584 CC Utrecht, The Netherlands
email: m.lopez@uu.nl

[2]Nikhef, Science Park 105, 1098 XG, Amsterdam, The Netherlands

[3]STAR Institute, Bâtiment B5, Université de Liège, Sart Tilman B4000 Liège, Belgium

[4]Department of Physics, University of Massachusetts, Dartmouth, MA 02747, USA

[5]Center for Scientific Computing and Visualization Research, University of Massachusetts, Dartmouth, MA 02747, USA

Abstract. In the field of gravitational-wave (GW) interferometers, the most severe limitation to the detection of transient signals from astrophysical sources comes from transient noise artefacts, known as glitches, that happens at a rate around 1 per minute. Because glitches reduce the amount of scientific data available, there is a need for better modelling and inclusion of glitches in large-scale studies, such as stress testing the search pipelines and increasing the confidence of detection. In this work, we employ a Generative Adversarial Network (GAN) to produce a particular class of glitches (*blip*) in the time domain. We share the trained network through a user-friendly open-source software package called `gengli` and provide practical examples of its usage.

Keywords. Generative adversarial networks, gravitational waves, synthetic data, machine learning.

1. Introduction

The existence of gravitational-wave (GW) signals was successfully proven by LIGO and Virgo collaborations during the first observing run (O1) (B. P. Abbott *et al.* 2016). After an upgrade of the detectors to increase their sensitivity, Advanced LIGO (J. Aasi *et al.* 2015) started in November 2016 the second observing run (O2), which Advanced Virgo (F. Acernese *et al.* 2015) joined in August 2017 (B. P. Abbott *et al.* 2017). Following significant upgrades, in April 2019, the third observing run (O3) was initiated by LIGO-Virgo collaboration (B. P. Abbott *et al.* 2020; R. Abbott *et al.* 2021). In the coming years, the improved second generation of interferometers and the construction of the third generation of detectors, such as Einstein Telescope (S. Hild *et al.* 2011; M. Maggiore *et al.* 2020), will increase significantly the detection sensitivity.

Despite the significant improvements to isolate the detectors from non-cosmic disturbances, they are still susceptible to non-Gaussian noise, known as "glitches", which come in a wide variety of time-frequency morphologies and are produced by instrumental or environmental causes. They reduce the amount of analyzable data, biasing astrophysical detection, and mimicking GW signals (B. P. Abbott *et al.* 2016), as in Fig. 1. Thus, it is fundamental to identify and characterize them for their elimination.

© The Author(s), 2025. Published by Cambridge University Press on behalf of International Astronomical Union.

https://doi.org/10.1017/S1743921323001047 Published online by Cambridge University Press

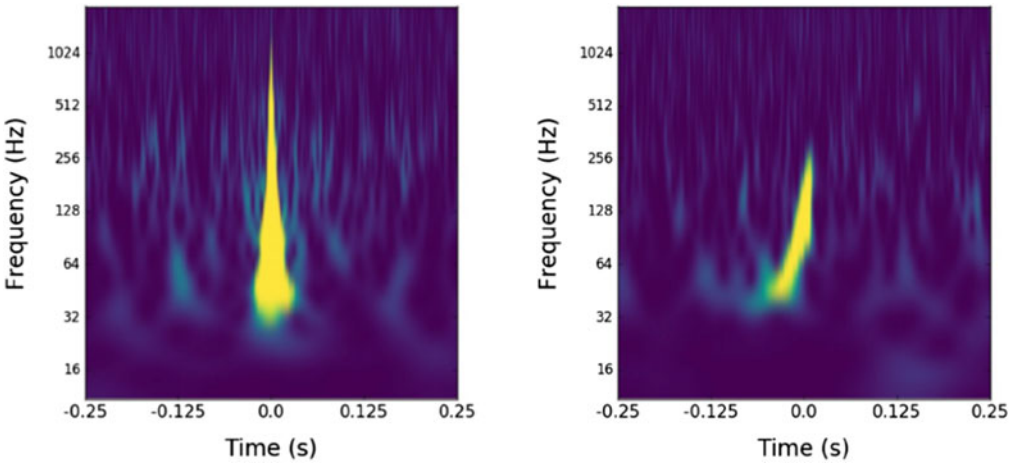


Figure 1. A blip glitch *(left)* that is similar to an event with total mass $106.6^{+13.5}_{-14.8} M_{\odot}$ *(right)*.

Due to the overwhelming amount of glitches present in the LIGO data (R. Abbott *et al.* (2021)), identifying them goes beyond human ability. An exciting solution is to construct machine learning (ML) algorithms to classify their different morphologies. With this idea in mind, M. Zevin *et al.* (2017) combine the strengths of both humans and computers to analyze and characterize LIGO glitches. Through the Zooniverse platform, volunteers provide large labelled data sets to train an ML algorithm, called *Gravity Spy*, while the ML algorithm learns to classify the rest of the glitches correctly and provides feedback to the participants. In practice, we feed to the ML classifier the glitch time series that we wish to classify. The Q-transform of the input is created and fed to the ML classifier, that assigns a label and a confidence value c_{GS}, where c_{GS} is the classification probability (see M. Zevin *et al.* (2017)).

While the identification of glitches is the first step towards their robust mitigation, in this investigation we intend to *generate* the known classes of glitches with Generative Adversarial Networks (GAN) to further understand their principal features. This does not only allow us to enhance our understanding of their morphologies but it can also be used for various applications in GW data analysis, such as mock data challenges (see M. Lopez *et al.* (2022) for a discussion). In this work, that accompanies the main publication M. Lopez *et al.* (2022), we present a flexible and user-friendly tool *gengli*† for glitch generation.

2. Data

To train our ML model, we select blips from Livingston (L1) and Hanford (H1) that have a confidence $c^1_{GS} \geq 0.9$, during O2‡. However, since the glitch is surrounded by non-stationary and uncorrelated noise, there is little structure that our ML method can retrieve. Therefore, we need to extract glitches from the input strain. For this aim, we employ BayesWave (N. J. Cornish and T. B. Littenberg 2015) to fit and reconstruct the input signal with data-driven wavelet models. The *whitening* operation is defined on the frequency domain glitch $\tilde{g}$ as: $\tilde{g}_W = \tilde{g}/S_n(f)$, where $S_n(f)$ is the power spectral density (PSD) of the input data.

The input provided to BayesWave is a time series of 2.0 s at 4096 Hz, where the blip is centred. After the reconstruction, we crop the output to a size of 938 data points,

† The code is released as a Git repository in melissa.lopez/gengli

‡ The data is accessible through the GWOSC (`https://www.gw-openscience.org/data/`)

https://doi.org/10.1017/S1743921323001047 Published online by Cambridge University Press

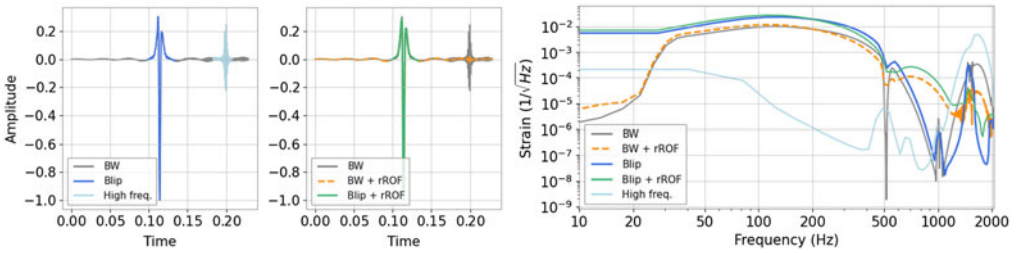


Figure 2. *(Left)* Reconstructed blip with BayesWave. *(Middle)* Denoised reconstructed blip with rROF. *(Right)* Resulting amplitude spectral density (ASD) for reconstructed and denoised blips.

removing irrelevant data. We evaluate the quality of the reconstruction by injecting the output in real whitened noise, and re-evaluating with Gravity Spy to select blip glitches with $c^2_{GS} \geq 0.9$.

The reconstruction of BayesWaves is not perfect since there is still some noise contribution at high frequencies (light blue) that will hinder the learning of our ML algorithm, (see Fig. 2 *(left)*). To minimize this contribution we employ regularized Rudin-Osher-Fatemi (rROF) proposed in A. Torres *et al.* (2014), which solves the denoising problem as a variational method. In Fig. 2 (middle), we plot the BayesWaves reconstruction denoised with rROF (dashed orange), and the denoised characteristic blip (green). In Fig. 2 (right), we show the amplitude spectral density (ASD) of the BayesWaves 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 maintain the structure of the characteristic peak by damping the power of the high-frequency contribution.

3. Methodology

3.1. *Generative Adversarial Networks*

GANs, first introduced by I. J. Goodfellow *et al.* (2014), are a class of generative algorithms in which two networks compete against each other to achieve realistic data generation. One network is responsible for the generation of new data from random noise, while the other tries to discriminate the generated samples from the real training data. The generator learns the features of real data that should be mimicked to "foul" the discriminator, and stores them in a latent space. At the end of the training, new samples are drawn by randomly taking a latent space vector and passing it to the generator, which translates this into real data.

The work from I. J. Goodfellow *et al.* (2014) suffers from two major problems, known as *vanishing gradients* and *meaningless loss function* (L. Weng 2019). To address those issues M. Arjovsky *et al.* (2017) developed Wasserstein GANs, where they use the Earth's mover distance or Wasserstein-1 distance (W_1) as a loss function. This change led to reformulate the optimization problem as $\theta_{opt} = arg\,\min_{\theta} W_1(P_x \| P_{\tilde{x}})$, where W_1 is evaluated between the true data distribution P_x and the fake data distribution $P_{\tilde{x}}$. Rewriting this equation yields,

$$\theta_{opt} = arg\,\min_{\theta} \max_{\phi:||D(x,\phi)||_L \leq 1} L(\phi,\theta) \quad \text{with } L(\phi,\theta) = -E_{x\sim P_x}\left[D(x,\phi)\right] + E_{\tilde{x}\sim P_{\tilde{x}}}\left[D(\tilde{x},\phi)\right] \qquad (1)$$

where $D(\phi)$ and $G(\theta)$ are for the discriminator and the generator is seen as a function of their weight. $E_{x\sim P_x}$ indicates that the expression has been averaged over a batch of real images. Similarly, $E_{\tilde{x}\sim P_{\tilde{x}}}$ indicates that the expression has been average over

https://doi.org/10.1017/S1743921323001047 Published online by Cambridge University Press

a batch of generated images, where $\tilde{x} = G(z, \theta)$ and z is the latent space vector. The new condition in Eq. 1 imposes that the discriminator must be 1-Lipschitz continuous (M. Arjovsky *et al.* 2017).

To fulfil this constraint, we implement the idea by I. Gulrajani *et al.* (2017), which consists in adding a regularization term to the discriminator loss, known as *gradient penalty* (GP):

$$L_{tot} = L(\phi, \theta) + \lambda\, GP(\phi)\; with\; GP(\phi) = E_{\hat{x}\sim P_{\hat{x}}}\left[\left(||\bigtriangledown_x\ D(\hat{x}, \phi)||_2 - 1\right)^2\right], \tag{2}$$

where λ is a regularization parameter, $||\cdot||_2$ denotes the L2-norm and $\hat{x}$ is evaluated following: $\hat{x} = \tilde{x}\, t + x\,(1-t)$, being x a real sample and $\tilde{x}$ a fake sample, with t sampled uniformly $\in [0, 1]$. However, GP cannot penalize W_1 everywhere. In particular, at the beginning of the training, being the generated samples quite far from the true data manifold, the Lipschitz condition is not enforced until the generator becomes sufficiently good.

To overcome this obstacle, X. Wei *et al.* (2018) have proposed a second penalization term that directly penalises the points near any observed real data point x. Therefore, when the gradient penalty fails to enforce the Lipschitz continuity in the close vicinity of x, the new term will constrain the latter. To penalize the real data manifold, X. Wei *et al.* (2018) applied their new constraint to two perturbed versions of the real samples x. For this, they introduced dropout layers into the discriminator architecture. The features kept by the dropout layer are random, which ultimately leads to two different estimates noted $D(x')$ and $D(x'')$. The penalty, called *consistency term* (CT), is then applied to these two estimates following:

$$CT(\phi) = E_{x\sim P_x}\left[max\left(0,\ d(D(x', \phi), D(x'', \phi)) + 0.1\, d(D_{\text{-}}(x', \phi), D_{\text{-}}(x'', \phi)) - M'\right)\right] \tag{3}$$

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. The final discriminator loss in X. Wei *et al.* (2018) is then:

$$L_D = L(\phi, \theta) + \lambda_1\, GP(\phi) + \lambda_2\, CT(\phi) \tag{4}$$

with λ_2 being the consistency parameter, that is used to tune the weight of CT for GP. While in Eq. 4 we update the weights of the discriminator, we update the weights of the generator as in M. Arjovsky *et al.* (2017), with the following expression:

$$L_G = -E_{\tilde{x}\sim P_{\tilde{x}}}\left[D(\tilde{x}, \phi)\right] \tag{5}$$

3.2. *Architecture and training procedure*

To build our model we use convolutional neural networks (CNN) in one dimension. In the generator, we make use of nearest-neighbour upsampling layers to avoid artefacts. We set the kernel size $k = 5$, padding $p = 0$, stride $s = 1$, with increasing dilation to enlarge the receptive field. Batch normalization is applied after each layer except for the penultimate one. Finally, $LeakyReLU(\cdot, \alpha = 0.2)$ activation function is employed, except in the output layer where we use a $Tanh(\cdot)$ activation, to constraint the output in the range $[-1, 1]$.

The discriminator structure is composed of strided convolutions on which spectral normalization is used, as suggested in I. Gulrajani *et al.* (2017). Dropout layers are triggered after each layer to regularise the discriminator, except for the first

https://doi.org/10.1017/S1743921323001047 Published online by Cambridge University Press

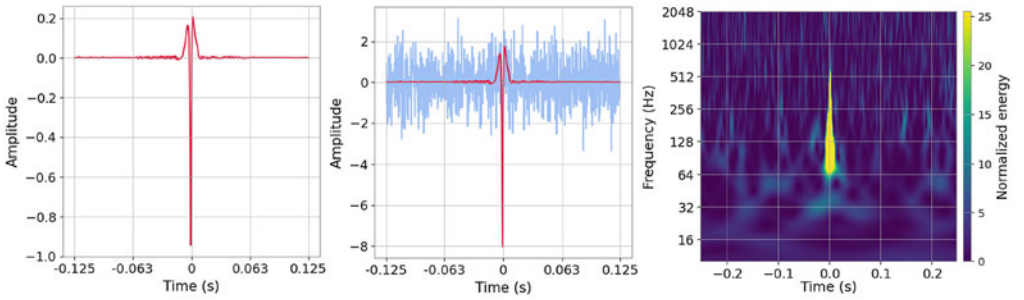


Figure 3. *(Left)* Raw blip as a function of time. *(Middle)* Raw blip injected in real (whitened) noise. *(Right)* Q-transform representation of injected glitch. Note the resemblance between the "fake" glitch here and the real glitch in Fig. 1.

and last layers. The kernel size is set to 5 for all layers, with no padding and $LeakyReLU(\cdot, \alpha = 0.2)$ activation.

During the training, both the generator and the discriminator need to be updated at similar rates for stability and convergence. However, as the classification task is more challenging, we update the discriminator 5 times per generator update. We employ RMSProp optimizer with a learning rate $= 10^{-4}$ for both discriminator and generator, and we train it for 500 epochs. Moreover, we employed $\lambda_1 = 5$, $\lambda_2 = 5$, and dropout rate of 0.6 for convergence. More details about the architecture and the training can be found in M. Lopez *et al.* (2022).

4. Implemented features

Once the model is trained, it is straightforward to use the generator network to produce random glitches. The output of the generator, a raw glitch, is the excess power of a *whitened* time series evaluated on a fixed length time grid at a constant sampling rate 4096 Hz and an amplitude in the range $[-1, 1]$. We can generate a raw glitch in ~ 10 ms on a laptop. A raw glitch can be straightforwardly injected into any whitened time series. In fig. 3 we plot an example of a raw glitch and of how it can be injected in white noise.

Since the amplitude of the generated glitches is arbitrary, before injecting the glitch into noise, a glitch has to be scaled to a user-given Signal-To-Noise (SNR) ratio ρ. The SNR of a glitch $g(t)$ is defined as:

$$\rho^2 = 4 \int_{f_{\min}}^{f_{\max}} \mathrm{d}f \frac{|\tilde{g}(f)|^2}{S_n(f)} \tag{6}$$

where again $\tilde{g}(f)$ is the glitch in the frequency domain and $S_n(f)$ is the actual PSD of the (whitened) data where the glitch is injected. The raw glitch is then scaled to achieve the target SNR ρ_{target} using: $g(t)' = \frac{\rho_{target}}{\rho} g(t)$

The raw glitch can also be naturally *resampled*§ at the desired sampling rate. All such functionalities are implemented in the function `get_raw_glitch`: the interested user can find below an example of some working code.

```
import gengli

g = gengli.glitch_generator('H1')
glitch = g.get_raw_glitch(snr = 10, srate = 2048.)
```

https://doi.org/10.1017/S1743921323001047 Published online by Cambridge University Press

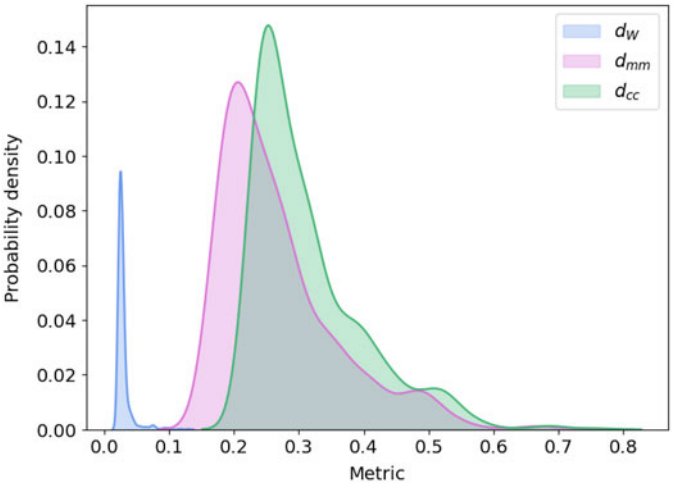


Figure 4. Probability distribution of distances (d_W, d_{mm}, d_{cc}) for a benchmark population of 10^3.

4.1. *Selecting generations*

In M. Lopez *et al.* (2022), it was shown that the generated glitches have the same statistical properties as the training population. However, the training data may (and most likely do) contain anomalous glitches that our model has also learnt to generate (see Section IV of M. Lopez *et al.* (2022)). In this work, we propose a novel method to filter anomalous glitches.

To make this intuition mathematically precise, we start by considering three different measures of distances between an arbitrary pair of glitches:

• Wasserstein distance d_W: a standard measure of distance between distributions, commonly employed in ML.

• Mis-match d_{mm}: a measure of distance based on the details of the filtering, standard in the GW field.

• Cross covariance d_{cc}: we employ the quantity $1 - k$, where k is the normalized cross-covariance as defined in M. Lopez *et al.* (2022).

We then populate a benchmark set of N_b glitches with samples from the generator. For each of the $\frac{N_b(N_b-1)}{2}$ pairs of glitches in the benchmark set, we compute the three distances above. In Fig. 4 we show the distribution for the three distances for a population of $N_b = 1000$.

For each new glitch being generated, we compute the set of average distances (d_W, d_{mm}, d_{cc}) between the glitch and the benchmark set, and we measure the set of percentiles (p_W, p_{mm}, d_{cc}) against the benchmark distances. The triplet (p_W, p_{mm}, d_{cc}) is our novelty measure for the glitch that will be used to filter glitches based on an anomaly score interval $[p_{min}, p_{max}]$. The code will output only glitches for which *all* the three anomaly scores lie within the interval.

5. Conclusions

We develop a methodology to generate artificial time-domain blip glitches data with the GAN algorithm. Because of the heavy pre-processing required to deal with real glitches (i.e. denoising, reconstructing etc...), this is a valid way to avoid using real data in state-of-the-art applications. Due to the instability of GAN algorithms, in this particular research, we trained the model constructed in I. Gulrajani *et al.* (2017), modified for time

§ When upsampling, we make the key assumption that there are no interesting features at frequencies higher than 4096 Hz.

https://doi.org/10.1017/S1743921323001047 Published online by Cambridge University Press

series, with the modified Wasserstein loss proposed in X. Wei *et al.* (2018). This model has shown better performance in both training stability and accuracy.

The performance of the generated blips has been assessed in the companion paper M. Lopez *et al.* (2022), where we employ several similarity distances: Wasserstein distance (d_W), mismatch (d_{mm}) and cross-covariance (d_{cc}). The results of these metrics indicate that our model was able to learn the underlying distribution of blip glitches despite the presence of anomalies due to imperfections of the input data set.

In this follow-up work, we introduce our open-source package `gengli`: it provides an easy-to-use interface to the trained GAN output, and has some additional features such as *resampling* and/or *scaling* a glitch and building a population with a given degree of "anomaly".

Future work will condition GAN to other types of glitches (such as *koi-fish* and *tomte*), by using a wealth of data from O3. Our work will enable the GW community to improve glitch classification with ML, study the properties of the glitch population and develop more realistic Mock Data challenges for studies on future GW detectors.

Acknowledgment

V.B. is supported by the Gravitational Wave Science (GWAS) grant funded by the French Community of Belgium, and M.L., S.C and S.S. are supported by the research program of the Netherlands Organisation for Scientific Research (NWO). Computational resources were provided by the LIGO Laboratory and supported by the National Science Foundation Grants No. PHY-0757058 and No. PHY-0823459.

References

J. Aasi *et al.* [LIGO Scientific], Class. Quant. Grav. **32**, 074001 (2015)
B. P. Abbott *et al.* [LIGO Scientific and Virgo], Class. Quant. Grav. **33**, no.13, 134001 (2016)
B. P. Abbott *et al.* [LIGO Scientific and Virgo] Phys. Rev. Lett. **116** no.6, 061102 (2016)
B. P. Abbott *et al.* [LIGO Scientific and Virgo], Phys. Rev. Lett. **119**, no.14, 141101 (2017)
B. P. Abbott *et al.* [LIGO Scientific and Virgo], Astrophys. J. Lett. **892**, no.1, L3 (2020)
R. Abbott *et al.* [LIGO Scientific, VIRGO and KAGRA], [arXiv:2111.03606 [gr-qc]].
F. Acernese *et al.* [VIRGO], Class. Quant. Grav. **32**, no.2, 024001 (2015)
M. Arjovsky, S. Chintala and L. Bottou (2017), arXiv:1701.07875
N. J. Cornish and T. B. Littenberg, Class. Quant. Grav. **32**, no.13, 135012 (2015)
I. J. Goodfellow *et al.*, arXiv:1406.2661.
I. Gulrajani *et al.*, arXiv:1704.00028 (2017)
S. Hild *et al.*, Class. Quant. Grav. **28**, 094013 (2011)
M. Lopez *et al.*, Phys. Rev. D **106**, no.2, 023027 (2022)
M. Maggiore *et al.*, JCAP 03, arXiv:1912.02622 (2020)
A. Torres, A. Marquina, J. A. Font and J. M. Ibáñez, Phys. Rev. D **90**, no.8, 084029 (2014)
X. Wei *et al.* (2018), arXiv:1803.01541
L. Weng, arXiv:1904.08994 (2019)
M. Zevin *et al.*, Class. Quant. Grav. **34**, no.6, 064003 (2017)

https://doi.org/10.1017/S1743921323001047 Published online by Cambridge University Press