# Generalised gravitational burst searches with Generative Adverserial Networks CHRIS: Need to change the title if we don't search[*]

(Dated: March 5, 2020)

As the field of gravitational wave astronomy expands and as detectors become more sensitive, there is a need for fast and efficient detection of various sources. Gravitational wave bursts are a family of signals that remain unmodeled, therefore, they have a low detection sensitivity to standard detection schemes. This report examines the use of Generative Adversarial Networks as a detection pipeline for gravitational wave burst signals. BurstGAN is trained to detect three classes of burst signals: sine-Gaussian, Ring-downs and white-noise bursts contained in white Gaussian noise. The resulting sensitivity curves show that sine-Gaussian and Ring-downs are confidently detectable at SNRs ¿ while white-noise bursts require SNRs of ¿. **CHRIS: we'll fix the abstract at the end**

## I. INTRODUCTION

Gravitational-wave (GW) astronomy is now an established field, starting with the first detection of a binary black hole merger [] on September 2015. Following this, the first and second observations runs (O1 and O2) of Advanced LIGO and Advanced Virgo reported several more mergers []. On August 2017 a binary neutron star merger was observed alongside its electron-magnetic counterpart for the first time, giving rise to multimessenger gravitational wave astronomy.

GW bursts are transient signals of typically short duration ($< 1$s) whose waveforms are not accurately modelled or are complex to re-produce. Astrophysical sources for such transients include: **JORDAN: should i include a list of these?**. Since GW bursts are un-modelled they are not sensitive to template based detection schemes such as matched-filtering [], instead, detection involves distinguishing the signal from detector noise. This is only possible if the detector noise is well characterised and the candidate signal can be differentiated from system or environmental glitches. As such, GW burst searches rely on an astrophysical burst signature appearing in multiple detectors. **JORDAN: how deep should i go? with coherent WaveBurst etc?**

Many GW burst algorithms [**?** ] are tested and tuned using model waveforms that may or may not have astrophysical significance but have easy to define parameters and share characteristics of real bursts that is enough to simulate coincident non-stationary deviations between detectors. Such waveforms may have long-duration, short bandwidth (ringdowns), long-duration, large bandwidth (inspirals) and many algorithms make use of sine-Gaussians: a Gaussian modulated sine wave that is characterised by it's central frequency and narrow bandwidth. This makes it a great tool for diagnosing LIGOs sensitivity to frequency.

We aim to explore the use of machine learning in generating and interpreting these mock GW burst signals. Neural networks have shown to replicate the sensitivities of matched filtering in GW detection [] and rapid parameter estimation [], however these methods have primarily been focused on binary black hole signals and have not yet expanded to burst examples. A subset of deep learning that has seen fruitful development in recent years [] is Generative Adversarial Networks (GANs). These unsupervised algorithms learn patterns in a given training data set using an adversarial process. The generations from GANs are state-of-the-art in fields such as high quality image fidelity, image and text-to-image translation and video prediciton [] and there is notable works on time series generations /cite.

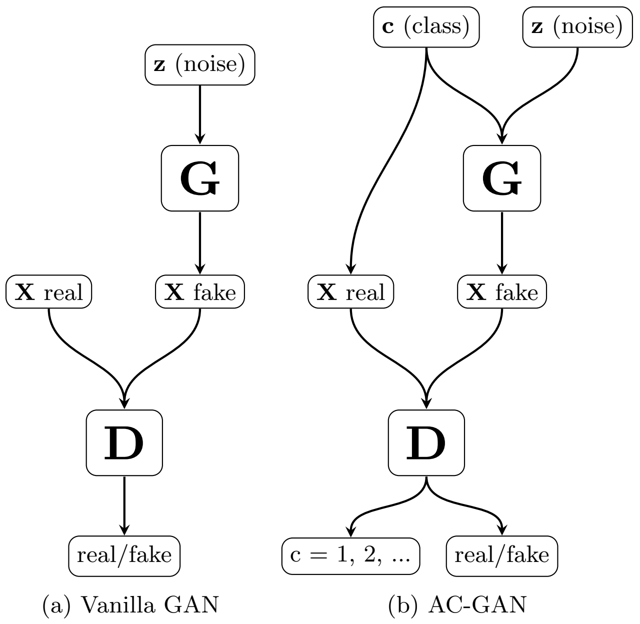

FIG. 1: Comparison of the original GAN method and the Auxiliary Conditional-GAN method

## II. GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) have seen considerable success in recent years as a generative machine learning model. GANs work to train two competing neural networks, consisting of a discriminator **D** that is trained to distinguish between real and fake data and a

[*] A footnote to the article title; Also at Physics Department, XYZ University.;

generator **G** that produces synthetic reproductions of the real data. The GAN method has **G** perform a mapping from an input noise vector **z** to its representation of the data and **D** maps its input **x** to a probability that the input came form either the training data or **G**. During training, **D** is given a batch of samples that contains one half real data (labelled as 1) and one half fake data (labelled as 0) which it then makes predictions on. The loss for **D** is calculated by comparing its predictions to the labelled data through the binary cross-entropy function. The training process of a GAN alternatively updates the weights of the **D** and **G** based on information on its competitors loss function. This loss of **D** is used to update the weights of **G** to produce more realistic samples of the input distribution, the loss of G encourages **D** to update its classification abilities. Both networks compete in a minimax game that is tied to a value function V (D, G) which **G** is trying minimise and **D** is trying to maximise:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{r}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{z}(\mathbf{z})}[\log(1-D(G(\mathbf{z})))] \quad (1)$$

### A. Auxiliary conditional GANs

In theory this will eventually lead to the local Nash equilibrium [18] where both neural networks are trained optimally. In practice, however, GANs are notoriously difficult to train. Such difficulties include: Non-convergence, where the model parameters oscillate and the loss never converges. Mode collapse where G produces a limited diversity of samples, and the diminishing gradient problem when applying gradient descent to a non-continuous function.
To overcome some of these difficulties [blank] proposed adding structure to the latent space by providing G with a class label. This has the effect of making a point in latent space conditional on a provided class. [blank] extends this idea further by requiring D to output a probability of the data belonging to each class.

## III. METHOD

- Need to introduce the scheme you propose to use
- A paragraph or subsection on the data generation being very clear on all 5 waveform models and the prior parameter space for each
- A subsection on the design of the network architecture
- A subsection on the "box" and why we implement it
- A subsection on the training of the network - give rough timings and rule of thumb decisions made

images/real_fake_sample.png

FIG. 2: Caption

- Do not discuss the results here

To simulate gravitational burst sources, three GW signal morphologies, spanning a range of frequencies, duration and time delays were tested. The three families are:

time delay/antenna pattern/lalsuite etc

archetecture/hyperparameters

## IV. RESULTS

- Begin by outlining the type of results you will be presenting
- A subsection on the general quality of generated waveforms - we may need to have overlaps between generated wavefoms and training data (maybe)
- A subsection on the descriminator - maybe a confusion matrix?
- a subsection on the latent space varaition within each class - fixed class, sliding in latent space.
- A subsection on the class space variation - fixed latent space and sliding in the class space.
- A final subsection on the general waveform model based on random latent and class space locations.
- Make no conclusions.

## V. CONCLUSIONS

- Summarise the paper
- Dedicate a paragraph to each of the key results discussed in the previous section
- Have at least one paragraph on the future directions of this work
- Conclude with a positive paragrpah about the potential uses and impact of the approach.

| Operation | Kernel | Strides | Output Shape | BN | Dropout | Activation |
|---|---|---|---|---|---|---|
| G($\mathbf{z}$): Input $\mathbf{z} \sim$ Normal(0,0.02) | N/A | N/A | (100,) | ✗ | 0 | N/A |
| Dense | N/A | N/A | (32768,) | ✗ | 0 | ReLU |
| Class input c | N/A | N/A | (1,) | ✗ | 0 | N/A |
| Embedding | N/A | N/A | (1, 120) | ✗ | 0 | N/A |
| Dense | N/A | N/A | (1,128) | ✗ | 0 | ReLU |
| Reshape $\mathbf{z}$ | N/A | N/A | (128, 256) | ✗ | 0 | N/A |
| Reshape c | N/A | N/A | (128, 1) | ✗ | 0 | N/A |
| Concatenate | N/A | N/A | (128, 257) | ✗ | 0 | N/A |
| Reshape | N/A | N/A | (64, 514) | ✗ | 0 | N/A |
| Transposed Convolution | 18x1 | 2 | (256, 256) | ✓ | 0 | ReLU |
| Transposed Convolution | 18x1 | 2 | (512, 128) | ✗ | 0 | ReLU |
| Transposed Convolution | 18x1 | 2 | (1024, 64) | ✗ | 0 | ReLU |
| Convolution | 18x1 | 1 | (1024, 1) | ✗ | 0 | Tanh |
| Sky input | N/A | N/A | (3,) | ✗ | 0 | N/A |
| Concatenate | N/A | N/A | (1027,) | ✗ | 0 | N/A |
| Lambda | N/A | N/A | (1024, 2) | ✗ | 0 | N/A |
| D($\mathbf{x}$): Input $\mathbf{x}$ | N/A | N/A | (1024, 2) | ✗ | 0 | N/A |
| Convolution | 14x1 | 2 | (512, 64) | ✗ | 0.5 | Leaky ReLU |
| Convolution | 14x1 | 2 | (256, 128) | ✗ | 0.5 | Leaky ReLU |
| Convolution | 14x1 | 2 | (128, 256) | ✗ | 0.5 | Leaky ReLU |
| Convolution | 14x1 | 2 | (64, 512) | ✗ | 0.5 | Leaky ReLU |
| Flatten | N/A | N/A | (32768,) | ✗ | 0 | N/A |
| Dense | N/A | N/A | (1,) | ✗ | 0 | Sigmoid |
| Dense | N/A | N/A | (5,) | ✗ | 0 | Softmax |
| Optimizer | Adam($\alpha = 0.0002$, $\beta_1 = 0.5$) | | | | | |
| Batch size | 100 | | | | | |
| Iterations | 60000 | | | | | |
| Leaky ReLU slope | 0.2 | | | | | |
| Weight initialization | Gaussian($\mu = 0$, $\sigma = 0.02$) | | | | | |
| Generator loss | Binary cross-entropy | | | | | |
| Discriminator loss | Binary cross-entropy & sparse categorical cross-entropy | | | | | |