Generalised gravitational burst generation with Generative Adversarial Networks

J. McGinn, C. Messenger, I.S. Heng, M. J. Williams
University of Glasgow, Physics & Astronomy Department, Glasgow G12 8QQ, UK

Abstract. The next generation of Gravitational wave detectors will accelerate the number of gravitational wave detection's such that we can gain new in site into the physics behind the sources causing the phenomena. Numerical simulations and matched filtering are the standard for detecting gravitational waves for know sources such as binary-black hole mergers. Other sources of gravitational waves that remain elusive to standard modelling techniques and are expected to be detectable, however, there must be a way to characterise them in order to build a model template. Here we construct a unmodeled burst generation scheme using Generative adversarial networks - a powerful class of machine learning.

1. Introduction

Gravitational wave astronomy is now an established field, starting with the first detection of a binary black hole merger [1] in September 2015. Following this, the first and second observations runs (O1 and O2) of Advanced LIGO and Advanced Virgo [2, 3, 4, 5] reported several more Compact Binary Coalescence (CBC) mergers [6, 7, 8, 9]. On August 2017 a binary neutron star merger was observed alongside its electromagnetic counterpart for the first time, giving rise to multimessenger gravitational wave astronomy.

With these successes and continued upgrades to the detectors, further detections of CBCs are expected to be commonplace in a few short years. Another group of GW signals that has thus far been undetectable is GW "bursts". GW bursts are transient signals of typically short duration (< 1s) whose waveforms are not accurately modelled or are complex to re-produce. Astrophysical sources for such transients include: Core collapse supernova [10], Pulsar glitches [11], Neutron star post-mergers [12] and other as-yet unexplained physical phenomena.

GW detections so far have used a process called matched filtering, [13], where a large template bank of possible GW waveforms are compared to the detector outputs. In order to increase the chances of detection these template banks must span a large multi dimensional parameter space and the templates must be accurate, requiring significant computational cost. GW bursts are un-modelled; therefore there are no templates available and so these signals are undetectable through this scheme. Instead, detection

involves distinguishing the signal from detector noise. Burst searches look for excess power contained in the time-frequency domain and rely on the astrophysical burst waveform appearing in multiple detectors at similar times. This is only possible if the detector noise is well characterised and the candidate signal can be differentiated from systematic or environmental glitches.

Gravitational wave (GW) burst algorithms [14, 15] 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 a GW passing between detectors. Such waveforms include sine-Gaussians: a Gaussian modulated sine wave that is by it's central frequency and decay parameter. Bandlimited white-noise bursts: white noise that is contained within a certain frequency range and ring-downs which mimic the damped oscillations after a CBC merger.

With the expectation that there will be many more GW detections in the future, researches are starting to realise the need for fast and efficient GW analysis to compete with the rising number of detections. While still in its infancy, Machine Learning (ML) with GW has already shown great potential for overcoming data bottlenecks. GW analysis in areas of detection [16, 17, 18], glitch classification [19, 20, 21] and parameter estimation [22, 23, 24] have shown the success of in ML pattern recognition. Applying these algorithms show similar results to matched filtering techniques and have the benefit of being orders of magnitude faster.

In this work we aim to explore the use of machine learning to generate and interpret unmodelled GW burst waveforms. Using the generative machine learning model: Generative Adversarial Networks (GANs), we train on five classes of known burst morphologies in the time domain. Working on the assumption that GANs construct smooth n-dimension vector spaces between it's input and output, we can then explore the space between the five classes to construct new unmodelled waveforms. As all the computationally expensive processes occur during training, the learned model after training is able to produce waveforms much faster than and in fact produce waveforms that are impossible to produce with current techniques. These new varieties of waveforms can then be used to diagnose detection algorithms and gain new insight into sources of GW bursts.

This paper is organised as follows: **JORDAN: unstructured at the moment**

2. Generative Adversarial Networks

2.1. Artificial neural networks

Since the birth of programmable computers, people have wondered if machines can develop true intelligence beyond formal mathematical procedures. Today, ML aims to learn apparent relationships held within the given data or 'training data' in order to make accurate predictions without the need for additional programming . A common approach in ML relies on the computer learning on past experience to make decisions

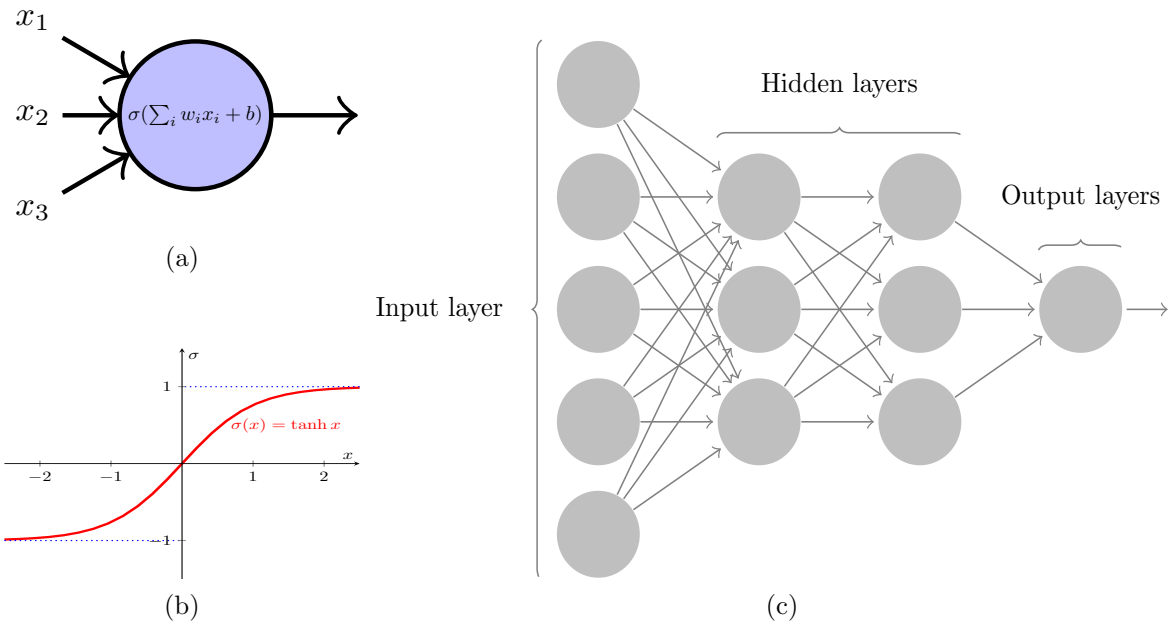

Figure 1: Neural Networks (a) A single neuron taking a vector of inputs and returning a singular output based on the weights, bias and activation function of the network. (b) The hyperbolic tangent used as an activation function. (c) A fully connected neural network containing two hidden layers that performs a mapping of an input vector to a singular output. **MICHAEL: Equation in the neuron looks very small**

on future events. The hierarchical nature of this approach means that the machine can build complicated concepts from simpler ones. This branch of AI is called deep learning. Neural networks are the quintessential machine learning algorithm that aims to approximate a function. They are built from many single processing units called neurons. The simpliest Neural Network is the perceptron layer Fig. 1a which holds a single neuron that takes several real inputs $x_1, ..., x_i$ and maps them to an output according to the following linear equation:

$$\sigma(\sum_i w_i x_i + b) \tag{1}$$

where w and b are the weights and bias and σ denotes the activation function. The weights are numbers which can be thought as the strength between connections. The output of a neuron is called its activation. Different choices of activation functions allow the user to simulate various models. A common example of an activation function is the hyperbolic tangent Fig. 1b that is used when the outputs of the network must be both positive and negative and re-scaled to $[-1, 1]$. It is often useful to introduce a bias, b, such that the neuron remains inactive above zero but is active when the sum reaches a defined threshold. This bias is added before the activation function and it tells us how high the weighted sum needs to be before the neuron becomes active.

The output of a single neuron is defined by Eq. (1) and gives a prediction $\hat{y}$ that

can be compared to the real value y through a cost or loss function. If the loss is non-zero, the network must work to minimise this function by updating the weights in the negative direction of the loss gradient in a process referred to as gradient descent. This loss function plays a critical role in how neural networks learn.

A neural network contains many single neurons connected in a layered structure as shown in Fig. 1c. The activations of the first layer (or input layer) act as the inputs to the second layer and so on until the output layer. Multilayered neural networks have intermediate layers between the input and output stages dubbed the hidden layers as the computations performed are not accessible to the user. The network can be thought of as a function F: $\mathbb{R}^N \rightarrow \mathbb{R}^M$ reliant on how activations from one layer bring activations in the next. The system is analogous to biology, where, some groups of neurons in animal brains cause certain others to fire.

2.2. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are designed to work with grid-like structures that exhibit strong local spatial dependencies. An example would be a two-dimensional image as each coloured pixel has similar intensities to their near neighbours. The value contained in each pixel can be thought of as adding an additional dimension or channel to the image. Although most work with CNNs involve image-based data, they can be applied to other spatially adjacent data types such as time-series and text items. CNNs are defined by the use of a convolution operation, a mathematical operation that expresses the amount overlap between functions. The convolution is applied by shifting one grid-like structure over the other, drawing out spatially important features between the two. A strided convolution reduces the dimensionality of the input while retaining spatial features. This can result in a loss of information along the boarders. To combat this, the input is padded with zeros at the edges which does not interfere with the convolutional operation. The output of the convolutional layer is then passed to an activation function.

2.3. GANs

A subset of deep learning that has seen fruitful development in recent years is generative adversarial networks (GANs) [25]. 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 [26, 27], text-to-image translation [28] and video prediction [29] as well as time series generations [30].

GANs train two competing neural networks, consisting of a discriminator that is set up to distinguish between real and fake data and a generator that produces synthetic reproductions of the real data. The generator performs a mapping from an input noise vector $\mathbf{z}$, that is usually sampled from a n-dimensional Gaussian space known as the latent space, to its representation of the data and the discriminator maps its input $\mathbf{x}$ to a probability that the input came form either the training data or generator. During

training, the discriminator is given a batch of samples that contains one half real data and one half fake data which it then makes predictions on. The loss for the discriminator is calculated by comparing its predictions to the labelled data through the binary cross-entropy function. **JORDAN: note to myself: This is important since eqn 2 is derived from it (thats where the logs come from)** The training process of a GAN alternatively updates the weights of the discriminator and generator based on information on its competitors loss function. This loss of discriminator is used to update the weights of the generator to produce more realistic samples of the input distribution, the loss of the generator encourages the discriminator to update its classification abilities. GANs seek to find an equilibrium between the generator and discriminator in the form of a two-player game that can be summarised by the following:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{r}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathrm{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2}$$

where, V is an objective function that is to be optimized. Objective functions are a general term for either a loss function (to be minimized) or a negative loss function (to be maximised). D is the discriminator, G is the generator and $\mathbf{x} \sim p_{\mathrm{r}}(\mathbf{x})$, $\mathbf{z} \sim p_{\mathrm{z}}(\mathbf{z})$ are samples taken from the training data and generated data respectfully. In practice this approach was found to be unstable. Early on in training the generators weights are initialised randomly meaning that the early generations are noisy and uninformative. This means that the discriminator can easily learn to spot the fake generations and so the discriminator wins and the generator cannot continue to improve. To overcome this the framing of the generator is often changed. Rather than minimizing $\log(1-D(G(\mathbf{z})))$ the objective now is to maximise $\log(D(G(\mathbf{z})))$, that is, to maximise the probability of the generations being predicted as real. The idea being that we always want the loosing side to be able to leverage from its competitor.

In theory, the adversarial process will eventually lead to the local Nash equilibrium [31] whereby 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 the generator produces a limited diversity of samples, and diminishing gradients when applying gradient descent to a loss function that is discontinuous.

2.4. Conditional GANs

To gain more control over what a GAN is able to generate, a conditional variant of GANs named conditional generative adversarial networks (CGANs) [32] was introduced by feeding in extra information into the generator and discriminator such as a class label or attribute label, c. The objective function is then modified:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x},\mathbf{c}\sim p_{\mathrm{r}}(\mathbf{x},\mathbf{c})}[\log D(\mathbf{x},\mathbf{c})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathrm{z}}(\mathbf{z}),\mathbf{c}\sim p_{\mathrm{r}}(\mathbf{c})}[\log(1 - D(G(\mathbf{z},\mathbf{c})))] \tag{3}$$

This simple addition has shown to work well in practice, for instance in image-to-image translation [33]. We will be using a conditional GAN for this study.

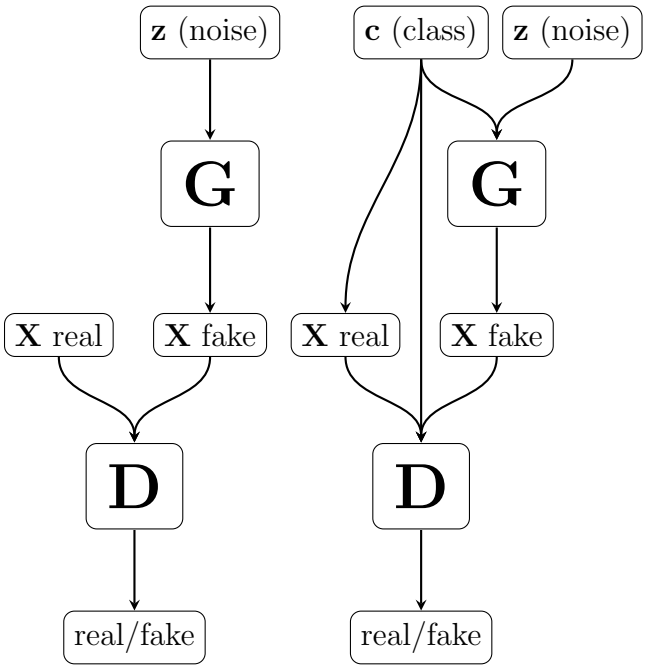

Figure 2: Comparison of the original GAN method and the Conditional-GAN method. For CGANs the training data requires a label denoting its class that is also fed to the generator which then learns to generate waveforms based on the input label.

3. Methodology

Table 1: Burst training parameters

Waveform	Central frequency (Hz)	Decay (s)	Central time epoch (s)	Mass range ($M_\odot$)
Sine-Gaussian	70 - 250	0.004 - 0.03	0.4 - 0.6	N/A
Ringdown	70 - 250	0.004 - 0.03	0.4 - 0.6	N/A
White-noise burst	70 - 250	0.004 - 0.03	0.4 - 0.6	N/A
Gaussian pulse	N/A	0.004 - 0.03	0.4 - 0.6	N/A
BBH	N/A	N/A	N/A	5 - 70

GW burst signals remain an unmodelled phenomenon, as such, current detection algorithms focus on waveforms signals appearing within multiple detectors We propose a signal generation scheme using GANs trained on burst-like waveforms. BurstGAN is a conditional GAN that is trained on five signal morphology's spanning a range of prior parameters. The parameter ranges can be seen in Table 1 The families are: sine-Gaussian $h(t) = A\exp\left[-(t-t_0)^2/\tau^2\right]\sin(2\pi f_0(t-t_0))$, a sinusoidal wave with a Gaussian envelop characterised by a central frequency, f_0; Ring-down, $h(t) = A\exp\left[-(t-t_0)/\tau\right]\sin(2\pi f_0(t-t_0))$, with frequency f_0 and duration τ, white-noise

bursts, with bounded frequency bandwidth Δf and duration τ, gaussian pulse $h(t) = \exp(-t^2/\tau^2)$ with duration τ and binary-black hole inspirals which are simulated using IMRPhenomD waveform [34] routine from LALSuite [35] which models the inspiral, merger and ringdown of a binary black hole (BBH) waveform. The component masses lie in the range of [5,70] $M_\odot$ with zero spins and we fix $m_1 > m_2$. The mass distribution is approximated by a power law with index of 1.6 [36]. The signals are generated using random right ascensions and declinations uniform over the sky and the inclinations are drawn from the cosine of a uniform distribution in the range [-1,1]. The peaks of the waveforms are set to be within [0.4,0.6]s of the 1s time interval.

All training waveforms are sampled at 1024 Hz.

3.1. Architecture details

Extensions to the original GAN method such as the deep convolutional generative adversarial network (DCGAN) [37] have been widely praised for enabling a stable GAN architecture. Most GANs now replace fully connected layers, where each neuron is connect to all the neurons in the previous and next layer, with convolutional layers. convolutional neural networks (CNN) are designed to work with grid-like structures with close local dependencies like image and text based data, however, there are examples of audio synthesis work [38].

We adopt the suggestions of [37, 38], lengthening one-dimensional convolution kernels on both the generator and discriminator. The Generator model is fully convolutional, upsampled using strided transposed convolutions with batch normalisation in the first layer and ReLU activations throughout with the exception of Tanh for the output layer. Each transposed convolutional layer uses a kernel size of 18×1 and stride of 2. The discriminator network mirrors that of the generator without batch normalization, using LeakyReLU activations, SpatialDropout, and a 2-stride convolution for downsampling. The discriminator output is a single node activated by a Sigmoid that can be interpreted as a probability of the the signal being real. This model is trained with binary cross entropy. The full architecture description can be seen in Table A1. **JORDAN: still too technical im not sure if we need to go into more detail on each thing mentioned.**

Neural networks and subsequently GANs have multiple parameters a developer can tune when designing the model and these are referred to as hyperparameters. The final network design used in this work comes from the use of trial and error and the initial designs influenced by the available literature. We found that the GAN performed better with both networks having the same number of layers and neurons which should encourage even competition between the generator and discriminator. After tuning the multiple hyperparamters (Table A1), the GAN was trained on 10^5 signals which contained an even mixture of sine-gaussian, ringdown, white noise bursts, gaussian blips and BBHs for 1000 epochs and takes O(2) days to train. A larger batch size saw a small increase in performance at the cost of training speed and we decided that 128 was a fair

compromise.

CHRIS: In general I find this section to read like un-connected statements in the sense that they are un-connected to each other and un-connected to the main GW problem. Try to make it more like a recipe for doing this analysis.

4. Results

Given a 100 dimensional vector drawn from a normal distribution, a class label and sky localisation information, the GAN is able to generate burst-like waveforms generalised from the training set. We set out by describing the quality of generated waveforms and how they compare to the training set. We then explore the structure of the latent and class spaces by interpolating between points in these spaces. We test vector arithmetic that can be used to generate a new breed of signal by merging two or more families together. Finally, we discuss the capacity of the discriminator as a GW burst classifier and the auxiliary component of this work.

4.1. Waveform quality

The generator network is a function $\mathrm{G} : \mathbf{z}, \mathbf{c}, \mathbf{s} \in \mathbb{R}^{100} \rightarrow \mathbb{R}^{1024\times 2}$, where $\mathbf{z}, \mathbf{c}, \mathbf{s}$ are the latent vector, class embedding vector and sky positions respectively. Given a latent vector randomly sampled from a normal distribution with zero mean and unit variance, a class label which is represented by a 5 dimensional one-hot encoded vector for each class, the results from the generator can be seen in Fig. 3. Each plot shows the output of the generator after given randomised $\mathbf{z}$ and one of the five class vectors $\mathbf{c}$.

4.2. Interpolation

Machine learning algorithms are often described as universal function approximators. In the generators case it maps samples drawn from a 100 dimensional Gaussian space to its representation of the training set. As with any function, there should be a one to one mapping from the domain and co-domain to allow for smooth transitions across the latent space. One advantage of using GANs as a waveform generator is that once it is trained, it can perform rapid generations faster than computationally expensive algorithms. For complicated data sets, the network architecture must be diverse and dense enough to capture distinct variations from the training set. Most GANs perform well on relatively low resolution image generations, however, higher resolutions demand larger networks and long training times. GANs attempting to replicate complicated structures and do not have the necessary architecture either struggle to produce results at all or fall into the common failure mode know as mode collapse; where the generator produces a small variety of samples or simply memorises the training set. To test this, we perform linear interpolations in the latent and class space.

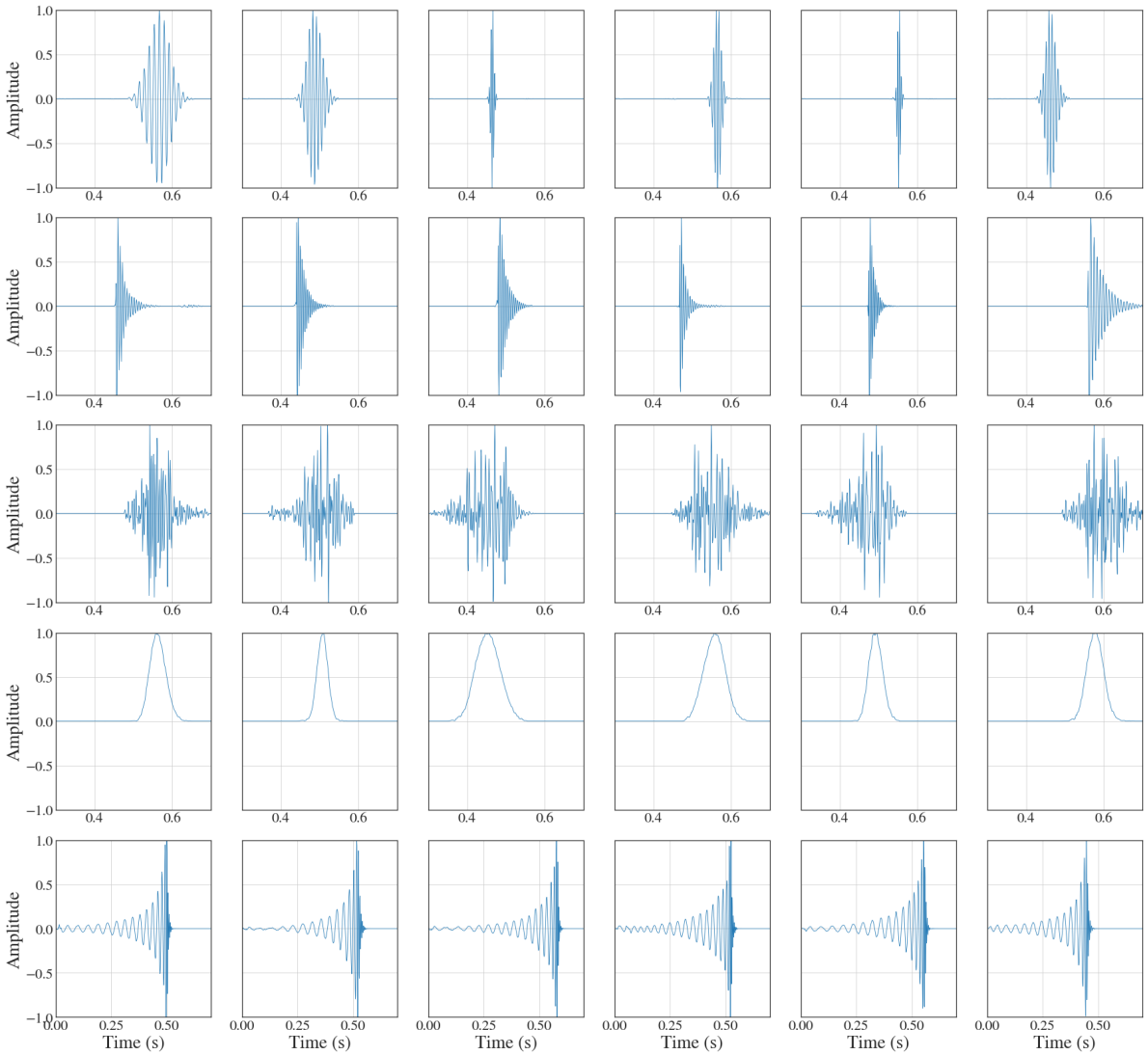

Figure 3: Examples of GW burst signals generated by a conditional generative adversarial network. By row: Sine-Gaussian, Ringdown, White-noise burst, Gaussian pulse, Binary black hole merger.

4.2.1. Latent space interpolation In this section we explore the latent space formed by the generator by interpolation. We take two random points in the latent space and linearly interpolate between them. These new latent space vectors can now be fed into the generator to make predictions on while keeping the class vectors constant. The full effect shown in Fig. 4. We can see that each plot shows plausible waveforms suggesting that the generator has constructed a smooth space unlike the discrete training case. Additionally as each class is given the same latent points to interpolate over, we can see that the waveforms cluster together with respect to their parameters. Visually, the sine-gaussian and ring-down waveforms share similar frequencies and the other signals show similar decays and starting epochs. The only exception is BBH waveforms, which is expected as they were trained with more variety of parameters and consistently have

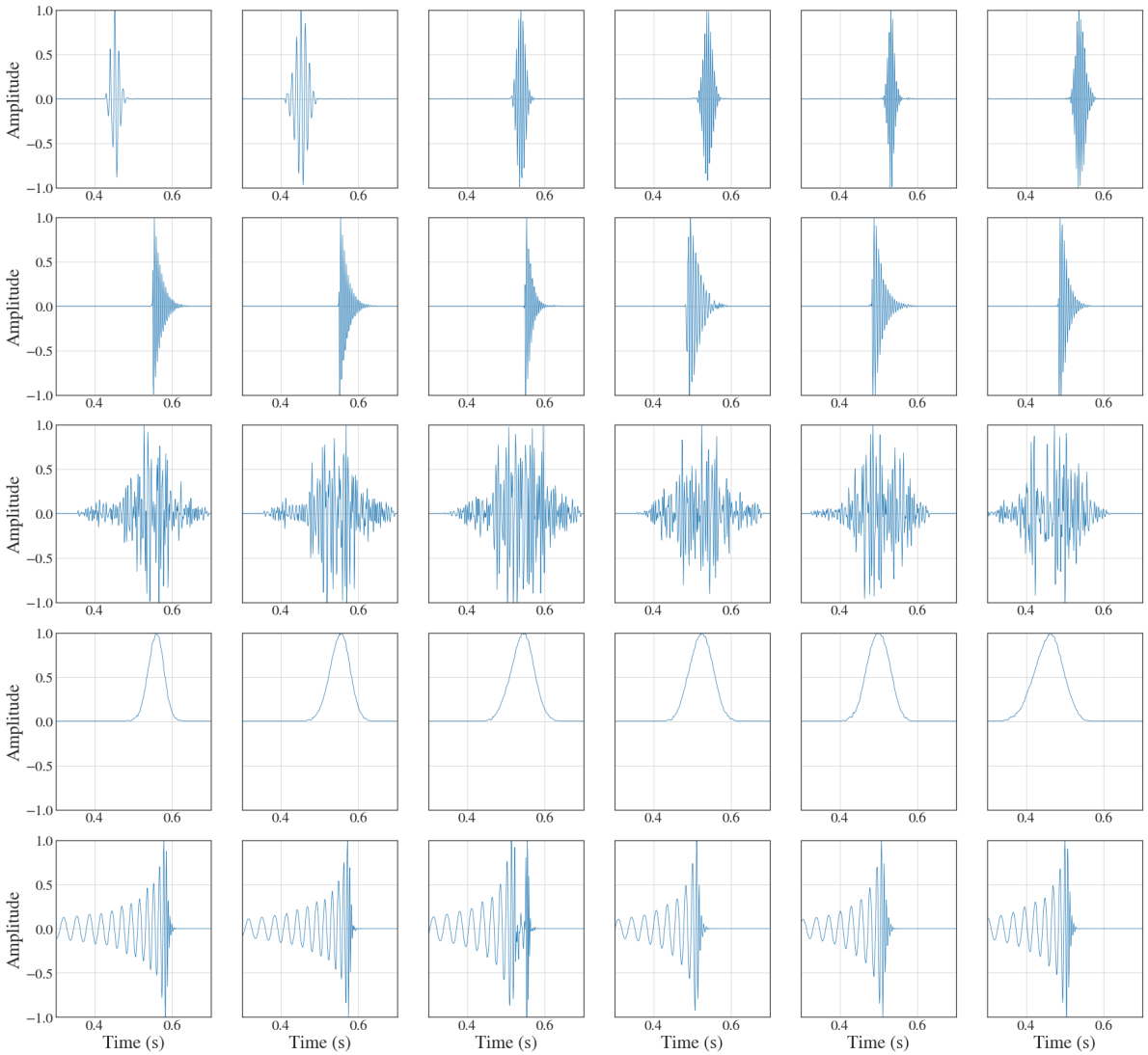

Figure 4: Latent space linear interpolation, class space is held constant.

their peaks in the last quarter of the time series.

4.2.2. Class space interpolation In order to explore the class space we keep the latent vector held constant and interpolate through the 5 classes. We construct a path between the 5 waveforms and show that the space is populated enough to allow for transitions between classes. Sine-Gaussian to ringdown performs well in interpolation with each signal being a plausible burst GW. It is obvious that the GAN has clustered these two groups during training as they share many characteristics. The other signals have sharper transitions but still retain plausible looking waveforms.

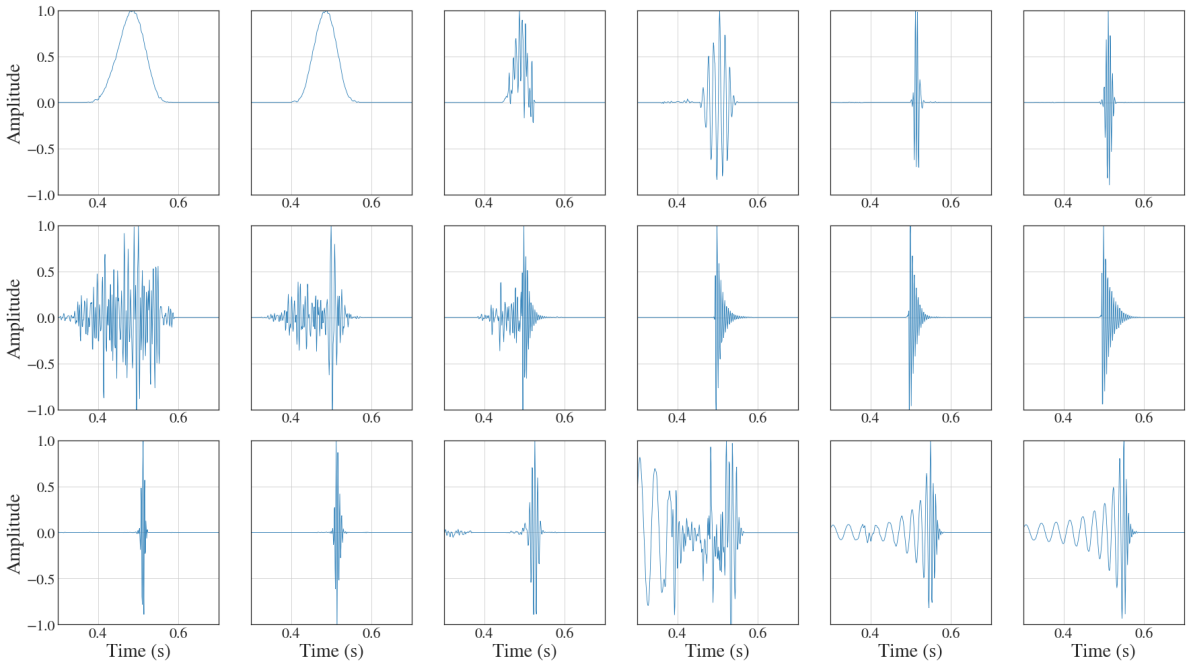

Figure 5: Class space linear interpolation, class space is held constant. Top row: gaussian blip to sine-gaussian, middle row: white noise burst to ringdown, bottom row: sine-gaussian to BBH.

4.3. Using a GAN to generate unmodelled waveforms

So far the analysis has focused on interpolating between two classes of signals with the aim to use the interpolated signals as unmodelled waveforms, instead, we may consider random mixtures of classes. We defined the classes in one-hot encoding framework, that is, each class resides at the corner points of a 5 dimensional cube. In order to generate an even mixture of classes we sample points uniformly in this space. **JORDAN: this would be the box**. Alternativly we can sampe points from the plane that intersects the four corners, which for a 5-dimensional case is, sample from a 4-simplex. Thks can nbee seen in 6. **JORDAN: why? this enforces the vector points to sum to one. Why is that good? Well it keeps the points lying on the simplex.**

4.4. CNN analysis

In this section of the analysis we will forensically compare generations from the GAN to help determine the success and failures of the model. We investigate a simple case of using a CNN to detect whether a signal is contained in noise or if the data is noise only. The signals we train the CNN on are generated from the GAN with their latent input randomised from a 100-dimensional Gaussian distribution and are categorised by their method of defining their class vector:

- Conditional – Definite categorical generations from the GAN. These signals are the closest to the training set.

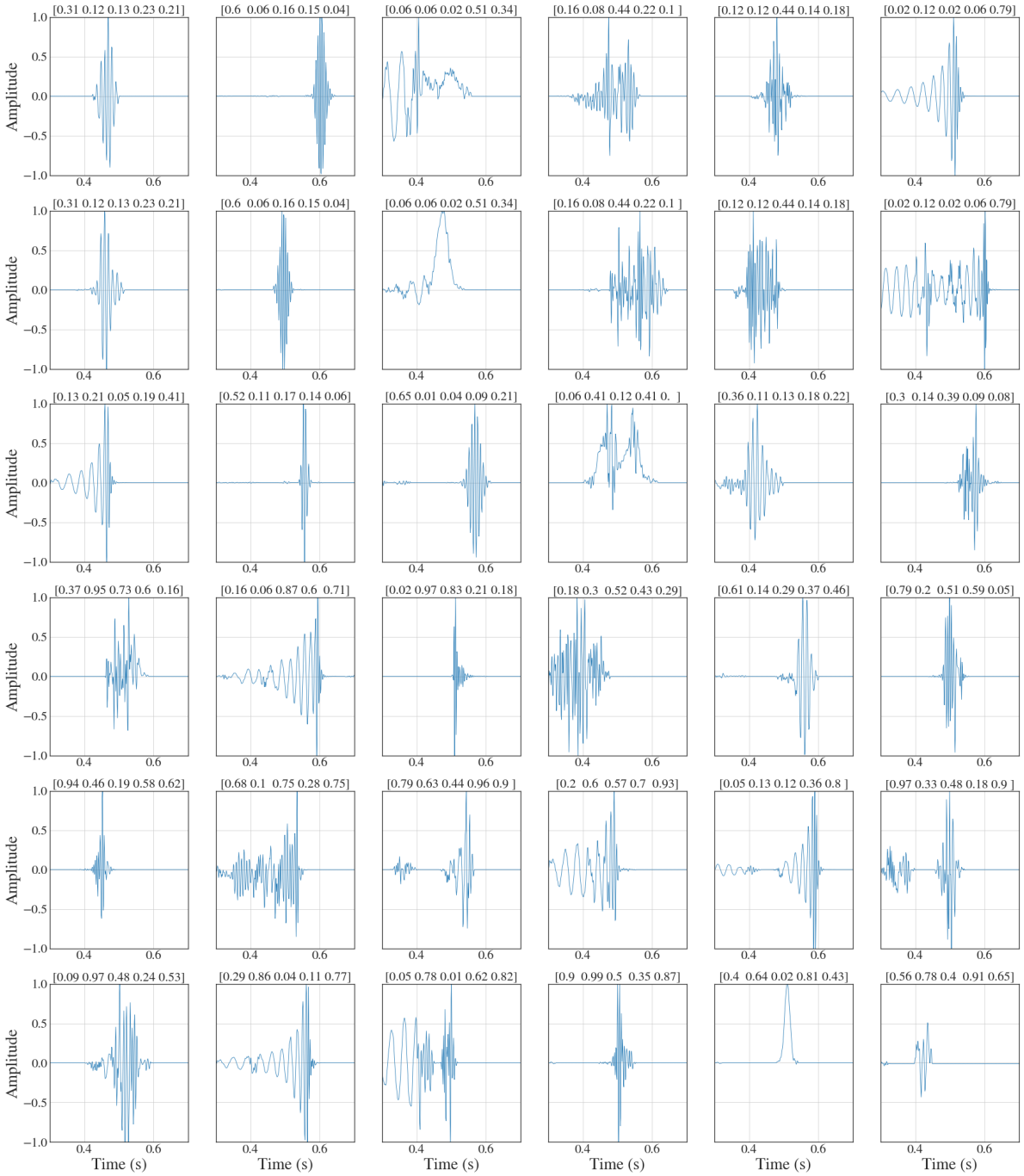

Figure 6: GAN generations where the class vectors are sampled from the 4-D plane and sampled uniformly in the class space.

- Uniform – Generated using a uniform distribution U[0,1] as the input class vector.
- Simplex – Generated using a Dirichlet distribution that samples from the 4-simplex as the input class vector.

The CNN is trained to distinguish between two classes: signals in additive Gaussian noise and Gaussian noise, where "signals" are taken from either the Conditional, Uniform or Simplex cases. This means in total there are three CNNs to train and all CNNs share the same architectural structure. All of the training data used is "whitened" by a power spectral density (PSD) at Advanced LIGO design sensitivity, such that, there is equal power at each frequency and is correctly normalised. This procedure is applied to signals whose signal-to-noise ratios (SNRs) are defined prior to be in the range uniformly [1-16]. For each run the training data consists of 200,000 signals which contain one half noise only and one half signal contained in noise. Of that 80% is used for training and 20% used for validation. A different testing set is also used in analysis that is 200,000 in size.

table of parameters used for CNN

In Fig. 7 we compare the CNN results between the three datasets, we train three CNNs on the Conditional, simplex and uniform data sets and use these models to make predictions on the other unseen datasets. We make a comparison by fixing the fraction of samples incorrectly identified as signals (false alarm rate) and plotting this versus the optimal SNR of the signals.

These results show for a CNN trained on conditional signals, the uniform data set is distinguishable from the conditional and simplex dataset. The CNN is robust enough to capture the differences between the datasets and shows that the GAN can generate a variety of unmodelled waveforms that can be used in future testing.

5. Conclusions

In this work we present the potential of Generative Adversarial Networks for burst gravitational wave analysis. We have shown that GANs have the ability to generate 5 class varieties of modelled burst GW signals that can be generated at whim. The latent and class spaces were explored through interpolation and suggest that the space provides smooth translations between classes and overall waveform shape. We then showed targeted waveform generation by mixing classes to produce new unmodlled waveform varieties that can be used to test current burst search pipelines. **JORDAN: couple of sentences about classifier**.

In order to extend this work to a viable burst wave generator and classifier a few points require further research. In principle it is trivial to add another detector inside the response layer **JORDAN: response is the wrong thing to say since the time delay isnt really a response, extrinsic layer? just non trainable layer? The box?**, however, as this now 3 dimensional signal is feed to the discriminator this will no doubt require further tweaking of the network. We can add more burst-like wave forms in the training set, like detector glitches which would similarly require further network design. The work presented here is noise free. To havea complete generation

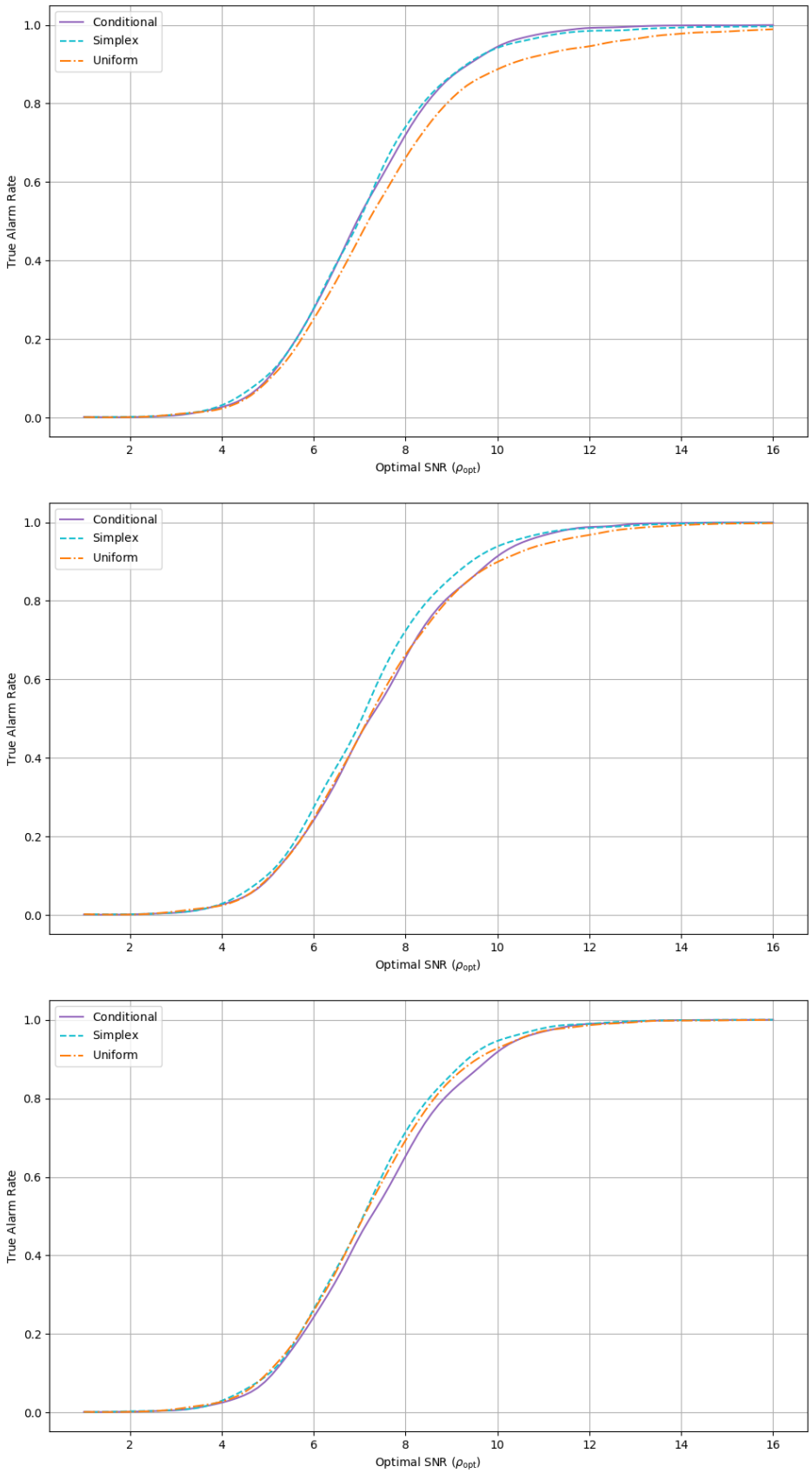

Figure 7: Efficiency curves comparing the performance of CNNs trained on conditional generations (top), simplex generations (middle), uniform generations (bottom) for a fixed false alarm rate of 10^{-3}.

and detection package we would like to train the network on signals hidden in additive Gaussian noise and test the ability of the auxiliary classifier.

The approach shown in this work shows promise in generating unmodelled burst waveforms from exotic sources. Having the ability to quickly generate new waveforms is essential to test current detection schemes and their susceptibilty to unmodelled sources. We belive that GANs have the ability to generate high fidelity waveforms at a fraction of the computational expense and do not rely on large prior parameter space. Having banks of these waveforms at hand can aid in our understand of the physics processes behind these non-standard gravitational wave emitters.

JORDAN: I want to include a link to the github etc but also a google collab scrip like the one for BigGANs: `https://colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/biggan_generation_with_tf_hub.ipynb`. **It's fun and gives a better feel for the interpolating that static images.**

References

[1] Abbott B *et al.* 2016 *Physical Review Letters* **116** 061102 ISSN 0031-9007
[2] Abbott B *et al.* (KAGRA, LIGO Scientific, VIRGO) 2018 *Living Rev. Rel.* **21** 3 (*Preprint* 1304.0670)
[3] Aasi J *et al.* (LIGO Scientific) 2015 *Class. Quant. Grav.* **32** 074001 (*Preprint* 1411.4547)
[4] Harry G M (LIGO Scientific) 2010 *Class. Quant. Grav.* **27** 084006
[5] Acernese F *et al.* (VIRGO) 2015 *Class. Quant. Grav.* **32** 024001 (*Preprint* 1408.3978)
[6] Abbott B *et al.* 2016 *Physical Review Letters* **116** 241103 ISSN 0031-9007
[7] Abbott B P *et al.* 2017 *The Astrophysical Journal Letters* **851** L35 ISSN 2041-8205
[8] Abbott B *et al.* 2017 *Physical Review Letters* **118** 221101 ISSN 0031-9007
[9] Abbott B *et al.* 2017 *Physical Review Letters* **119** 161101 ISSN 0031-9007
[10] Fryer C L and New K C B 2003 *Living Reviews in Relativity* **6** 2
[11] Andersson N and Comer G L 2001 *Phys. Rev. Lett.* **87**(24) 241101
[12] Baiotti L, Hawke I and Rezzolla L 2007 *Classical and Quantum Gravity* **24** S187–S206
[13] Owen B J and Sathyaprakash B S 1998 *Matched filtering of gravitational waves from inspiraling compact binaries: Computational cost and template placement* (*Preprint* 9808076)
[14] Klimenko S *et al.* 2008 *Classical and Quantum Gravity* **25** 114029
[15] Aso Y *et al.* 2008 *Classical and Quantum Gravity* **25** 114039
[16] Gabbard H *et al.* 2017 *Matching matched filtering with deep networks in gravitational-wave astronomy* (*Preprint* 1712.06041)
[17] Gebhard T D *et al.* 2019 *Physical Review D* **100** ISSN 2470-0029
[18] Krastev P G 2020 *Physics Letters B* **803** 135330 ISSN 0370-2693
[19] Bahaadini S *et al.* 2017 *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* pp 2931–2935
[20] George D, Shen H and Huerta E 2018 *Physical Review D* **97** ISSN 2470-0029
[21] Razzano M and Cuoco E 2018 *Classical and Quantum Gravity* **35** 095016
[22] Gabbard H *et al.* 2019 Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy (*Preprint* 1909.06296)
[23] Shen H *et al.* 2019 Deterministic and bayesian neural networks for low-latency gravitational wave parameter estimation of binary black hole mergers (*Preprint* 1903.01998)
[24] Green S R, Simpson C and Gair J 2020 Gravitational-wave parameter estimation with autoregressive neural network flows (*Preprint* 2002.07656)

[25] Goodfellow I *et al.* 2014 *Advances in Neural Information Processing Systems 27* ISSN 10495258
[26] Brock A, Donahue J and Simonyan K 2018 Large scale gan training for high fidelity natural image synthesis (*Preprint* 1809.11096)
[27] Karras T *et al.* 2019 Analyzing and improving the image quality of stylegan (*Preprint* 1912.04958)
[28] Reed S *et al.* 2016 Generative adversarial text to image synthesis (*Preprint* 1605.05396)
[29] Liang X *et al.* 2017 Dual motion gan for future-flow embedded video prediction (*Preprint* 1708.00284)
[30] Esteban C, Hyland S L and Rätsch G 2017 Real-valued (medical) time series generation with recurrent conditional gans (*Preprint* 1706.02633)
[31] Nash J F 1950 *Proceedings of the National Academy of Sciences* **36** 48–49 ISSN 0027-8424
[32] Mirza M and Osindero S 2014 *CoRR* **abs/1411.1784** (*Preprint* 1411.1784)
[33] Isola P *et al.* 2016 Image-to-image translation with conditional adversarial networks (*Preprint* 1611.07004)
[34] Khan S *et al.* 2016 *Physical Review D* **93** ISSN 2470-0029
[35] LIGO Scientific Collaboration 2018 LIGO Algorithm Library - LALSuite free software (GPL)
[36] Abbott B P *et al.* 2019 *The Astrophysical Journal* **882** L24 ISSN 2041-8213
[37] Radford A, Metz L and Chintala S 2015 *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks* (*Preprint* 1511.06434)
[38] Brock A, Donahue J and Simonyan K 2018 *CoRR* **abs/1809.11096** (*Preprint* 1809.11096)

Appendix A. List of hyperparameters

Table A1: ACGAN architecture

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	128					
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					

Appendix B. Many more generated examples