Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy

Hunter Gabbard[1],* Chris Messenger[1], Ik Siong Heng[1], Francesco Tonolini[2], and Roderick Murray-Smith[2]
SUPA, School of Physics and Astronomy[1], University of Glasgow, Glasgow G12 8QQ, United Kingdom

School of Computing Science[2], University of Glasgow, Glasgow G12 8QQ, United Kingdom

(Dated: September 18, 2020)

Gravitational wave (GW) detection is now commonplace [1, 2] and as the sensitivity of the global network of GW detectors improves, we will observe $\mathcal{O}(100)$s of transient GW events per year [3]. The current methods used to estimate their source parameters employ optimally sensitive [4] but computationally costly Bayesian inference approaches [5] where typical analyses have taken between 6 hours and 5 days [6]. For binary neutron star (BNS) and neutron star black hole (NSBH) systems prompt counterpart electromagnetic (EM) signatures are expected on timescales of 1 second − 1 minute and the current fastest method for alerting EM follow-up observers [7], can provide estimates in $\mathcal{O}(1)$ minute, on a limited range of key source parameters. Here we show that a conditional variational autoencoder (CVAE) [8, 9] pre-trained on binary black hole (BBH) signals can return Bayesian posterior probability estimates. The training procedure need only be performed once for a given prior parameter space and the resulting trained machine can then generate samples describing the posterior distribution ~ 6 orders of magnitude faster than existing techniques.

The problem of detecting GWs has largely been solved through the use of template based matched-filtering, a process recently replicated using machine learning techniques [10–12]. Once a GW has been identified through this process, Bayesian inference, known to be the optimal approach [4], is used to extract information about the source parameters of the detected GW signal.

In the standard Bayesian GW inference approach, we assume a signal and noise model and both may have unknown parameters that we are either interested in inferring or prefer to marginalise away. Each parameter is given a prior astrophysically motivated probability distribution and in the GW case, we typically assume a Gaussian additive noise model (in reality, the data is not truly Gaussian). Given a noisy GW waveform, we would like to find an optimal procedure for inferring some set of the unknown GW parameters. Such a procedure should be able to give us an accurate estimate of the parameters of our observed signal, whilst accounting for the uncertainty arising from the noise in the data.

According to Bayes' Theorem, a posterior probability distribution on a set of parameters, conditional on the measured data, can be represented as

$$p(x|y) \propto p(y|x)p(x), \tag{1}$$

where x are the parameters, y is the observed data, $p(x|y)$ is the posterior, $p(y|x)$ is the likelihood, and $p(x)$ is the prior on the parameters. The constant of proportionality, which we omit here, is $p(y)$, the probability of our data, known as the Bayesian evidence or the marginal likelihood. We typically ignore $p(y)$ since it is a constant and for parameter estimation purposes we are only interested in the shape of the posterior.

Due to the size of the parameter space typically encountered in GW parameter estimation and the volume of data analysed, we must stochastically sample the parameter space in order to estimate the posterior. Sampling is done using a variety of techniques including Nested Sampling [13–15] and Markov chain Monte Carlo methods [16, 17]. The primary software tools used by the advanced Laser Interferometer Gravitational wave Observatory (LIGO) parameter estimation analysis are `LALInference` and `Bilby` [5, 18], which offer multiple sampling methods.

Machine learning has featured prominently in many areas of GW research over the last few years. These techniques have shown to be particularly promising in signal detection [10–12], glitch classification [19], earthquake prediction [20], and to augment existing Bayesian sampling methods [21]. We also highlight recent developments in GW parameter estimation (independent to this work) where one- and two-dimensional marginalised Bayesian posteriors are produced rapidly using neural networks [22], and where normalised flows in conjunction with CVAEs can reproduce Bayesian posteriors for a single GW detector case [23]. These methods, including the one presented in this paper, are known as "likelihood-free" approaches in which there is no requirement for explicit likelihood evaluation [24], only the need to sample

from the likelihood. Nor is it the case that pre-computed posterior distributions are required in the training procedure.

Recently, a type of neural network known as CVAE was shown to perform exceptionally well when applied towards computational imaging inference [8, 25], text to image inference [26], high-resolution synthetic image generation [27] and the fitting of incomplete heterogeneous data [28]. CVAEs, as part of the variational family of inference techniques are ideally suited to the problem of function approximation and have the potential to be significantly faster than existing approaches. It is therefore this type of machine learning network that we apply in the GW case to accurately approximate the Bayesian posterior $p(x|y)$, where x represents the physical parameters that govern the GW signal, and are the quantities we are interested in inferring. The data y represents the noisy measurement containing the GW signal and obtained from a network of GW detectors.

The construction of a CVAE begins with the definition of a quantity to be minimised (referred to as a cost function). In our case we use the cross entropy, defined as

$$H(p, r) = -\int dx\, p(x|y) \log r_\theta(x|y) \tag{2}$$

between the true posterior $p(x|y)$ and $r_\theta(x|y)$, the parametric distribution that we will use neural networks to model and which we aim to be equal to the true posterior. The parametric model is constructed from a combination of 2 (encoder and decoder) neural networks $r_{\theta_1}(z|y)$ and $r_{\theta_2}(x|y, z)$ where

$$r_\theta(x|y) = \int dz\, r_{\theta_1}(z|y) r_{\theta_2}(x|y, z). \tag{3}$$

In this case the θ subscripts represent sets of trainable neural network parameters and the variable z represents locations within a *latent space*. This latter object is typically a lower dimensional space within which an encoder can represent the input data, and via marginalisation allows the construction of a rich family of possible probability densities.

Starting from Eq. 2 it is possible to derive a computable bound for the cross-entropy that is reliant on the r_{θ_1} and r_{θ_2} networks and a third "recognition" encoder network $q_\phi(z|x, y)$ governed by the trainable parameter-set ϕ. The details of the derivation are described in the methods section and in [8] but equate to an optimisation of the evidence lower bound (ELBO). The final form of the cross-entropy cost function is given by the bound

$$H \lesssim \frac{1}{N} \sum_{n=1}^{N_\mathrm{b}} \Big[\overbrace{-\log r_{\theta_2}(x_n|z_n, y_n)}^{L} + \overbrace{\mathrm{KL}\left[q_\phi(z|x_n, y_n)||r_{\theta_1}(z|y_n)\right]}^{\mathrm{KL}} \Big], \tag{4}$$

which is also represented graphically in Fig. 1. The cost function is composed of 2 terms, the "reconstruction" cost L which is a measure of how well the decoder network r_{θ_2} predicts the true signal parameters x, and the Kullback–Leibler (KL)-divergence cost that measures the similarity between the distributions modelled by the r_{θ_1} and q_ϕ encoder networks. In practice, for each iteration of the training procedure, the integrations over x, y and z are approximated by a sum over a batch of N_b draws from the user defined prior $p(x)$, the known likelihood $p(y|x)$, and the recognition function $q_\phi(z|, x, y)$. Details of the training procedure are given in the methods section.

The implementation of the CVAE that we employ in this letter has a number of specific features that were included in order to tailor the analysis to GW signals. The details of these enhancements are described in the Methods section but in summary, the primary modifications are as follows, 1) Physically appropriate output decoder distributions are used for each output parameter: von Mises-Fisher distribution on the sky location parameters, von Mises distributions on periodic parameters, conditional truncated Gaussians for the component masses, and truncated Gaussians for parameters with defined prior bounds. 2) Each of the functions $r_{\theta_1}, r_{\theta_2}$, and q_ϕ are modelled using deep convolutional neural networks with multi-detector time-series represented as independent input channels. 3) The r_{θ_1} encoder models an $M = 16$ component Gaussian mixture model within the $n_z = 10$ dimensional latent space in order to capture the corresponding typical multi-modal nature of GW posterior distributions.

We present results on 256 multi-detector GW test BBH waveforms in simulated advanced detector noise [29] from the LIGO Hanford, Livingston and Virgo detectors. We compare between variants of the existing Bayesian approaches and our CVAE implementation which we call `VItamin`. Posteriors produced by the `Bilby` inference library [18] are used as a benchmark in order to assess the efficiency and quality of our machine learning approach with the existing methods for posterior sampling.

For the benchmark analysis we assume that 9 parameters are unknown[1]: the component masses m_1, m_2, the luminosity distance d_L, the sky position α, δ, the binary inclination Θ_{jn}, the GW polarisation angle ψ, the time of coalescence t_0, and the phase at coalescence ϕ_0. For each parameter we use a uniform prior with the exception of the declination and inclination parameters for which we use priors uniform in $\cos\delta$ and $\sin\Theta_{jn}$ respectively. The corresponding prior ranges are defined in Table II and result in an signal-to-noise ratio (SNR) distribution

[1] Our analysis omits the 6 additional parameters required to model the spin of each BBH component mass.

FIG. 1. The configuration of the CVAE neural network. During training (left-hand side), a training set of noisy GW signals (y) and their corresponding true parameters (x) are given as input to encoder network q_ϕ, while only y is given to encoder network r_{θ_1}. The KL-divergence (Eq. 7) is computed between the encoder output latent space representations (μ_q and μ_r) forming one component of the total cost function. Samples (z_q) from the q_ϕ latent space representation are generated and passed to the decoder network r_{θ_2} together with the original input data y. The output of the decoder (μ_x) describes a distribution in the physical parameter space and the cost component L is computed by evaluating that distribution at the location of the original input x. When performed in batches this scheme allows the computation of the total cost function Eq. 4. After having trained the network and therefore having minimised the cross-entropy H, we test (right-hand side) using only the r_{θ_1} encoder and the r_{θ_2} decoder to produce samples ($x_{\rm samp}$). These samples are drawn from the distribution $r_\theta(x|y)$ (Eq. 3) and accurately model the true posterior $p(x|y)$.

that peaks at SNR ≈ 9 and ranging between 0 and 75. We use a sampling frequency of 256 Hz, a time-series duration of 1 second, and the waveform model used is `IMRPhenomPv2` [30] with a minimum cutoff frequency of 20Hz. For each input test waveform we run the benchmark analysis using multiple sampling algorithms available within `Bilby`. For each run and sampler we extract $\mathcal{O}(10^4)$ samples from the posterior on the 9 physical parameters.

The `VItamin` training process uses as input 10^7 whitened waveforms corresponding to parameters drawn from the same priors as assumed for the benchmark analysis. The waveforms are also of identical duration, sampling frequency, and use the same waveform model as in the benchmark analysis. The signals are whitened[2] using the same advanced detector PSDs [29] as assumed in the benchmark analysis. When each whitened waveform is placed within a training batch it is given a unique detector Gaussian noise realisation (after signal whitening this is simply zero mean, unit variance Gaussian noise). The `VItamin` posterior results are produced by passing each of our 256 whitened noisy testing set of GW waveforms as input into the testing path of the pre-trained CVAE (Fig. 1). For each input waveform we sample until we have generated 10^4 posterior samples on 7 physical parameters $x = (m_1, m_2, d_{\rm L}, t_0, \Theta_{jn}, \alpha, \delta)$. We choose to output a subset of the full 9-dimensional space to demonstrate that parameters (such as ϕ_0 and ψ in this case) can (if desired) be marginalised out within the CVAE procedure itself, rather than after training.

We can immediately illustrate the accuracy of our machine learning predictions by directly plotting 2 and one-dimensional marginalised posteriors generated using the output samples from our `VItamin` and `Bilby` approaches superimposed on each other. We show this for one example test dataset in Fig. 2 where strong agreement between 2 `Bilby` samplers (`Dynesty` in blue, and `ptemcee` in green) and the CVAE (red) is clear. It is also evident that whilst we refer to the `Bilby` sampler results as benchmark cases, different existing samplers do not perfectly agree with each other. For each of our 256 test cases we see equivalent levels of disparity between pairs of benchmark samplers *and* between any benchmark sampler and our CVAE results.

Figures 4 and 5 (see the Methods section) show the results of 2 statistical tests (the probability-probability (p-p) plot test and KL-divergence tests) performed on the entire test dataset and between all samplers (`Dynesty`, `ptemcee`, `CPNest`, `emcee`, and `VItamin`). In both tests the quality of the `VItamin` results are indistinguishable from the benchmark samplers. The p-p plot results specifically indicate that the Bayesian one-dimensional marginalised posteriors from each approach are self-consistent from a frequentist perspective (e.g., the true values lie within the $X\%$ confidence interval for $X\%$ of the test cases). The second test computes the distribution of KL-divergences between posteriors conditioned on the same test data y from pairs of samplers. In all cases this measure of "distribution similarity" between `VItamin` and any particular benchmark sampler is entirely consistent with the distribution between that benchmark sampler and any other.

The dominating computational cost of running

[2] The whitening is used primarily to scale the input to a magnitude range more suitable to neural networks. The *true* power spectral density (PSD) does not have to be used for whitening, but training data and test data must be contain signals that share the same PSD.

FIG. 2. Corner plot showing one and two-dimensional marginalised posterior distributions on the GW parameters for one example test dataset. Filled red contours represent the two-dimensional joint posteriors obtained from `VItamin` and solid blue and green contours are the corresponding posteriors output from our benchmark analyses (using the `Dynesty` and `ptemcee` samplers within `Bilby`). In each case, the contour boundaries enclose 68, 90 and 95% probability. One dimensional histograms of the posterior distribution for each parameter from both methods are plotted along the diagonal. Black vertical and horizontal lines denote the true parameter values of the simulated signal. At the top of the figure we include a Mollweide projection of the sky location posteriors from all three analyses. All results presented in this letter correspond to a three-detector configuration but for clarity we only plot the H1 whitened noisy time-series y and the noise-free whitened signal (in blue and cyan respectively) to the right of the figure. The test signal was simulated with an optimal multi-detector signal-to-noise ratio of 17.2.

TABLE I. Durations required to produce samples from each of the different posterior sampling approaches.

sampler	run time (seconds) min	max	median	ratio $\frac{\tau_{\texttt{VItamin}}}{\tau_X}$
`Dynesty`[a] [15]	11795	29838	19400 [b]	5.2×10^{-6}
`emcee` [16]	18838	69272	32070	3.1×10^{-6}
`ptemcee` [17]	17124	37446	24372	4.1×10^{-6}
`CPNest` [14]	9943	53315	26202	3.8×10^{-6}
`VItamin`[c]	$\mathbf{1 \times 10^{-1}}$			1

[a] The benchmark samplers all produced $\mathcal{O}(10000)$ samples dependent on the default sampling parameters used.

[b] We note that there are a growing number of specialised techniques [31–33] designed to speed up traditional sampling algorithms that could be used to reduce the runtimes quoted here by $\mathcal{O}(1-2)$ orders of magnitude.

[c] For the `VItamin` sampler 10000 samples are produced as representative of a typical posterior. The run time is independent of the signal content in the data and is therefore constant for all test cases.

`VItamin` lies in the training time, which takes $\mathcal{O}(1)$ day to complete. We stress that once trained, there is no need to retrain the network unless the user wishes to use different priors $p(x)$ or assume different noise characteristics. The speed at which posterior samples are generated for all samplers used, including `VItamin`, is shown in Table I. Run-time for the benchmark samplers is defined as the time to complete their analyses when configured using the parameter choices defined in Table IV. For `VItamin`, this time is defined as the total time to produce 10^4 samples. For our test case of BBH signals `VItamin` produces samples from the posterior at a rate which is ~ 6 orders of magnitude faster than our benchmark analyses using current inference techniques.

In this letter we have demonstrated that we are able to reproduce, to a high degree of accuracy, Bayesian posterior probability distributions generated through machine learning. This is accomplished using a CVAE trained on simulated GW signals and does not require the input of precomputed posterior estimates. We have demonstrated that our neural network model, which when trained, can reproduce complete and accurate posterior estimates in a fraction of a second, achieves the same quality of results as the trusted benchmark analyses used within the LIGO-Virgo Collaboration.

The significance of our results is most evident in the orders of magnitude increase in speed over existing algorithms. We have demonstrated the approach using BBH signals but with additional work to increase sample rate and signal duration, the method can also be extended for application to signals from BNS mergers (e.g., GW170817 [2], and GW190425 [34]) and NSBH systems where improved low-latency alerts will be especially pertinent. By using our approach, parameter estimation speed will no longer be limiting factor[3] in observing the prompt EM emission expected on shorter time scales than is achievable with existing LIGO-Virgo Collaboration (LVC) analysis tools such as Bayestar [7].

The predicted number of future detections of BNS mergers ($\sim$ 180 [3]) will severely strain the GW community's current computational resources using existing Bayesian methods. We anticipate that future iterations of our approach will provide full-parameter estimation on all classes of compact binary coalescence (CBC) signals in $\mathcal{O}(1)$ second on single graphics processing units (GPUs). Our trained network is also modular, and can be shared and used easily by any user to produce results. The specific analysis described in this letter assumes a uniform prior on the signal parameters. However, this is a choice and the network can be trained with any prior the user demands, or users can cheaply resample accordingly from the output of the network trained on the uniform prior. We also note that our method will be invaluable for population studies since populations may now be generated and analysed in a fully-Bayesian manner on a vastly reduced time scale.

For BBH signals, GW data is usually sampled at 1—4 kHz dependent upon the mass of binary. We have chosen to use the noticeably low sampling rate of 256Hz in order to decrease the computational time required to develop our approach and the computational burden of computing our 256 benchmark analyses for each of 4 benchmark samplers. We have been able to extend our analysis to 1 kHz sampling frequencies at the cost of an ~ 1.5 fold increase in training time and a similar increase on the GPU memory requirement. We note that with the exception of requiring one-dimensional convolutional layers and an increase in the amount of training data to efficiently deal with a multi-detector analysis, the network complexity has not increased with the dimensionality of the physical parameter space nor with the sampling rate of the input data. We therefore do not anticipate that extending the parameter space to lower masses and including component spin parameters will be problematic.

In reality, GW detectors are affected by non-Gaussian noise artefacts and time-dependent variation in the detector noise PSD. Existing methods incorporate a parameterised PSD estimation into their inference [35]. To account for these and to exploit the "likelihood-free" nature of the CVAE approach, we could re-train our network at regular intervals using samples of real detector noise (preferably recent examples to best reflect the state

[3] A complete low-latency pipeline includes a number of steps. The process of GW data acquisition is followed by the transfer of data. There is then the corresponding candidate event identification, parameter estimation analysis, and the subsequent communication of results to the EM astronomy community after which there are physical aspects such as slewing observing instruments to the correct pointing.

of the detectors). In this case we could also apply transfer learning to speed up each training instance based on the previously trained network state. Alternatively, since the PSD is an estimated quantity, we could marginalise over its uncertainty by providing training data whitened by samples drawn from a distribution of possible PSDs. Our work can naturally be extended to include the full range of CBC signal types but also to any and all other parameterised GW signals and to analyses of GW data beyond that of ground based experiments. Given the abundant benefits of this method, we hope that a variant of this of approach will form the basis for future GW parameter estimation.

ACKNOWLEDGEMENTS.

We would like to acknowledge valuable input from the LIGO-Virgo Collaboration, specifically from Will Farr, Tom Dent, Jonah Kanner, Alex Nitz, Colin Capano and the parameter estimation and machine-learning working groups. We would additionally like to thank Szabi Marka for posing this challenge to us and the journal referees for their helpful and constructive comments. We thank Nvidia for the generous donation of a Tesla V-100 GPU used in addition to LVC computational resources. The authors also gratefully acknowledge the Science and Technology Facilities Council of the United Kingdom. CM and SH are supported by the Science and Technology Research Council (grant No. ST/ L000946/1) and the European Cooperation in Science and Technology (COST) action CA17137. FT acknowledges support from Amazon Research and EPSRC grant EP/M01326X/1, and RM-S EPSRC grants EP/M01326X/1, EP/T00097X/1 and EP/R018634/1.

ADDENDUM

Competing Interests

The authors declare that they have no competing financial interests.

Correspondence

Correspondence and requests for materials should be addressed to Hunter Gabbard (email: h.gabbard.1@research.gla.ac.uk).

METHODS

A CVAE is a form of variational autoencoder that is conditioned on an observation, where in our case the observation is a one-dimensional GW time-series signal y. The autoencoders from which variational autoencoders are derived are typically used for problems involving image reconstruction and/or dimensionality reduction. They perform a regression task whereby the autoencoder attempts to predict its own given input (model the identity function) through a "bottleneck layer" — a limited and therefore distilled representation of the input parameter space. An autoencoder is composed of two neural networks, an encoder and a decoder [36]. The encoder network takes as input a vector, where the number of dimensions is a fixed number predefined by the user. The encoder converts the input vector into a (typically) lower dimensional space, referred to as the *latent space*. A representation of the data in the latent space is passed to the decoder network which generates a reconstruction of the original input data to the encoder network. Through training, the two sub-networks learn how to efficiently represent a dataset within a lower dimensional latent space which will take on the most important properties of the input training data. In this way, the data can be compressed with little loss of fidelity. Additionally, the decoder simultaneously learns to decode the latent space representation and reconstruct that data back to its original form (the input data).

The primary difference between a variational autoencoder [9] and an autoencoder concerns the method by which locations within the latent space are produced. In our variant of the variational autoencoder, the output of the encoder is interpreted as a set of parameters governing statistical distributions. In proceeding to the decoder network, samples from the latent space (z) are randomly drawn from these distributions and fed into the decoder, therefore adding an element of variation into the process. A particular input can then have a range of possible outputs. Any trainable network architectures can be used in both the decoder and the encoder networks and within `VItamin` we use deep convolutional neural networks in both cases.

Cost function derivation

We will now derive the cost function and the corresponding network structure and we begin with the statement defining the aim of the analysis. We wish to obtain a function that reproduces the posterior distribution (the probability of our physical parameters x given some measured data y). The cross-entropy between 2 distributions is defined in Eq. 2 where we have made the distributions explicitly conditional on y (our measurement). In this case $p(x|y)$ is the target distribution (the true posterior) and $r_\theta(x|y)$ is the parametric distribution that we will use neural networks to construct. The variable θ represents the trainable neural network parameters.

The cross-entropy is minimised when $p(x|y) = r_\theta(x|y)$

and so by minimising

$$H = -\mathrm{E}_{p(y)}\left[\int dx\, p(x|y)\log r_\theta(x|y)\right], \tag{5}$$

where $\mathrm{E}_{p(y)}[\cdot]$ indicates the expectation value over the distribution of measurements y, we therefore make the parametric distribution as similar as possible to the target for all possible measurements y.

Converting the expectation value into an integral over y weighted by $p(y)$ and applying Bayes' theorem we obtain

$$H = -\int dx\, p(x)\int dy\, p(y|x)\log r_\theta(x|y) \tag{6}$$

where $p(x)$ is the prior distribution on the physical parameters x, and $p(y|x)$ is the likelihood of x (the probability of measuring the data y given the parameters x).

The CVAE network outlined in Fig. 1 makes use of a conditional latent variable model and our parametric model is constructed from the product of 2 separate distributions marginalised over the latent space as defined in Eq. 3. We have used θ_1 and θ_2 to indicate that the 2 separate networks modelling these distributions will be trained on these parameter sets respectively. The encoder $r_{\theta_1}(z|y)$ takes as input the data y and outputs parameters that describe a probability distribution within the latent space. The decoder $r_{\theta_2}(x|z,y)$ takes as input a single location z within the latent space together with the data y and outputs sets of parameters describing a probability distribution in the physical parameter space.

One could be forgiven for thinking that by setting up networks that simply aim to minimise H over the θ_1 and θ_2 would be enough to solve this problem. However, as shown in [25] this is an intractable problem and a network cannot be trained directly to do this. Instead we introduce a recognition function $q_\phi(z|x,y)$, modelled by an additional neural network and governed by the trainable network parameters ϕ, that will be used to derive an ELBO.

Let us first define the KL-divergence between the recognition function and the distribution $r_\theta(z|x,y)$ as

$$\mathrm{KL}\left[q_\phi(z|x,y)||r_\theta(z|x,y)\right] = \int dz\, q_\phi(z|x,y)\log\left(\frac{q_\phi(z|x,y)}{r_\theta(z|x,y)}\right), \tag{7}$$

from which it can be shown that

$$\log r_\theta(x|y) = \mathrm{ELBO} + \mathrm{KL}\left[q_\phi(z|x,y)||r_\theta(z|x,y)\right], \tag{8}$$

where the ELBO is given by

$$\mathrm{ELBO} = \int dz\, q_\phi(z|x,y)\log\left(\frac{r_{\theta_2}(x|y,z)r_{\theta_1}(z|y)}{q_\phi(z|x,y)}\right). \tag{9}$$

It is so-named since the KL-divergence has a minimum of zero and cannot be negative. Therefore, if we were to find a $q_\phi(z|x,y)$ function (optimised on ϕ) that minimised the KL-divergence defined in Eq. 7 then we can state that

$$\log r_\theta(x|y) \geq \mathrm{ELBO}. \tag{10}$$

After some further manipulation of Eq. 9 we find that

$$\log r_\theta(x|y) \geq \mathrm{E}_{q_\phi(z|x,y)}\left[\log r_{\theta_2}(x|z,y)\right] - \mathrm{KL}\left[q_\phi(z|x,y)||r_{\theta_1}(z|y)\right]. \tag{11}$$

We can now substitute this inequality into our cost function as defined by Eq. 6 to obtain

$$H \leq -\int dx\, p(x)\int dy\, p(y|x)\Big[\mathrm{E}_{q_\phi(z|x,y)}\left[\log r_{\theta_2}(x|z,y)\right] - \mathrm{KL}\left[q_\phi(z|x,y)||r_{\theta_1}(z|y)\right]\Big], \tag{12}$$

which can in practice be approximated as a stochastic integral over draws of x from the prior, y from the likelihood function $p(y|x)$, and from the recognition function, giving us Eq. 4, the actual function evaluated within the training procedure. In standard sampling algorithms it is required that the likelihood is calculated explicitly during the exploration of the parameter space and hence an analytic noise and signal model must be assumed. For a CVAE implementation we are required only to sample from the likelihood distribution, i.e., generate simulated noisy measurements given a set of signal parameters. This gives us the option of avoiding the assumption of detector noise Gaussianity in the future by training the CVAE using "real" non-Gaussian detector noise.

TABLE II. The uniform prior boundaries and fixed parameter values used on the BBH signal parameters for the benchmark and the CVAE analyses.

Parameter name	symbol	min	max	units
mass 1	m_1	35	80	solar masses
mass 2	m_2[a]	35	80	solar masses
luminosity distance	d_L	1	3	Gpc
time of coalescence	t_0	0.65	0.85	seconds
phase at coalescence	ϕ_0	0	2π	radians
right ascension	α	0	2π	radians
declination	δ	$-\pi/2$	$\pi/2$	radians
inclination	ι	0	π	radians
polarisation	ψ	0	π	radians
spins	-	0		-
epoch		1126259642		GPS time
detector network		LIGO H1,L1, & Virgo V1		-

[a] Additionally m_2 is constrained such that $m_2 < m_1$.

Network design

The CVAE network outlined in Fig. 1 is constructed from the 3 separate neural networks modelling the encoder and decoder distributions r_{θ_1} and r_{θ_2} as well as

FIG. 3. The cost as a function of training iteration. We show the total cost function (green) together with its component parts: the KL-divergence component (orange) and the reconstruction component (blue) which are simply summed to obtain the total. The solid curves correspond to the cost computed on each batch of training data and the dashed curves represent the cost when computed on independent validation data. The close agreement between training and validation cost values indicates that the network is not overfitting to the training data. The change in behavior of the cost between 10^4 and 10^5 iterations is a consequence of gradually introducing the KL cost term contribution via an annealing process.

the recognition function q_ϕ. Each of these components is a deep convolutional network consisting of a series of one-dimensional convolutional layers followed by a series of fully-connected layers. The details of each network structure are given in Table III where we indicate the activations used and additional dropout and batch-normalisation layers.

The r_{θ_1} network takes the input time-series data y in the form of multi-channel 1-dimensional vectors where channels represent different GW detectors. After passing through a series of one-dimensional convolutional and fully connected layers, the output then defines the parameters of a n_z-dimensional (diagonal) Gaussian mixture model in the latent space. We label these parameters as μ_{r_1} containing $n_z = 10$ means and log-covariances together with the $M = 16$ mixture component weights. The motivation for using this mixture model representation comes from the multi-modal nature of GW posterior distributions. The encoder network can use this flexibility to represent the y time-series data as belonging to multiple possible latent space regions. Due to this added flexibility we note that the r_{θ_1} network is generally required to be more complex to learn this more complicated distribution, hence the additional layers in comparison to the other network components.

The recognition function network q_ϕ is very similar to the r_{θ_1} network with only 2 differences. The network takes as input the y time-series and the true signal parameters x, however, only the y data is passed through the one-dimensional convolutional layers. Only after the final convolutional layer where the output is flattened is the x data appended. It is then this compound time-series data "feature-space" and true signal parameters that are processed using the remaining fully-connected layers. The second difference is that the output of the network defines a *single-modal* (diagonal) n_z-dimensional Gaussian. We label these parameters as μ_q containing $n_z = 10$ means and log-covariances. The rationale behind this choice is that since the q_ϕ distribution is conditional on the true signal parameters, there should be no ambiguity as to which mode in the latent space that a particular time-series belongs to.

The decoder network r_{θ_2} is identical in structure to the q_ϕ network but with a difference in the form of their outputs. The r_{θ_2} output represents the parameters (μ_{r_2}) that govern an n_x-dimensional distribution in the physical parameter space and we have carefully chosen appropriate distributions for each of the physical parameters. For the luminosity distance, the binary inclination, and the time of coalescence we have adopted truncated Gaussian distributions where the truncation occurs at the predefined prior boundaries of the respective parameter space dimensions. For the component masses we have adopted conditional truncated Gaussians where the conditional aspect is to ensure that $m_1 \geq m_2$[4]. Had we chosen to explicitly infer the coalescence phase and polarisation angles, independent von Mises distributions would be applied to capture the periodic nature of these parameters. Finally, we use the von Mises-Fisher distribution to model the right ascension and declination (sky) parameters.

We automatically tune a subset of the network hyperparameters (one-dimensional convolutional filter size, filter stride length, max-pool size, number of modes (M)) through Bayesian optimisation using Gaussian Processes [37]. We use the `scikit-optimize` toolkit [38] and optimise on the total loss of our validation dataset. An initial set of network hyperparameters are randomly chosen according to a predefined uniform distribution. The network is then trained for $\sim 10^6$ iterations before the total validation loss is recorded. The Gaussian Process optimisation algorithm chooses a new set of network hy-

[4] This additional complication of requiring conditional decoder output distributions could be avoided if a different mass parameterisation were chosen, e.g., total mass and symmetric mass ratio.

TABLE III. The `VItamin` network hyper-parameters

Network / Layer	$r_{\theta_1}(z\|y)$	$r_{\theta_2}(x\|y,z)$	$q_\phi(z\|x,y)$
Input y	[256,3][a]	[256,3]	[256,3]
Layer 1	conv(5,3,33)[b] act[c]=ReLU	conv(5,3,33) act=ReLU	conv(5,3,33) act=ReLU
Layer 2	conv(8,33,33) maxpool(2,2)[d] act=ReLU	conv(8,33,33) maxpool(2,2) act=ReLU	conv(8,33,33) maxpool(2,2) act=ReLU
Layer 3	conv(11,33,33) act=ReLU	conv(11,33,33) act=ReLU	conv(11,33,33) act=ReLU
Input z, x	-	flatten[e]→[4224] append[f](z)→[4234]	flatten→[4224] append(x)→[4231]
Layer 4	conv(10,33,33) maxpool(2,2) act=ReLU	FC(4234,2048)[g] dropout(0.2)[h] act=ReLU	FC(4231,2048) dropout(0.2) act=ReLU
Layer 5	conv(10,33,33) act=ReLU flatten→[2112]	FC(2048,2048) dropout(0.2) act=ReLU	FC(2048,2048) dropout(0.2) act=ReLU
Layer 6	FC(2112,2048) dropout=0.2 act=ReLU	FC(2048,14) act=(Sigmoid,-ReLU)[i] output=μ_{r_2} →[7,2][j]	FC(2048,20) act=None output=μ_q →[10,2][k]
Layer 7	FC(2048,2048) dropout(0.2) act=ReLU		
Layer 8	FC(2048,320) act=None output=μ_{r_1} →[10,16,2][l]		

[a] The shape of the data [one-dimensional dataset length, No. channels].
[b] one-dimensional convolutional filter with arguments (filter size, No. channels, No. filters).
[c] The activation function used.
[d] Max-pooling layer with arguments (pool size, stride length).
[e] Take the multi-channel output of the previous layer and reshape it into a one-dimensional vector.
[f] Append the argument to the current dataset.
[g] Fully connected layer with arguments (input size, output size).
[h] Drop-out layer with argument (drop-out fraction).
[i] Different activations are used for different parameters. For the scaled parameter means we use sigmoids and for log-variances we use negative ReLU functions.
[j] The r_{θ_2} output has size [physical space dimension, No. parameters defining the distribution per dimension].
[k] The q_ϕ output has size [latent space dimension, No. parameters defining the distribution per dimension].
[l] The r_{θ_1} output has size [latent space dimension, No. modes, No. parameters defining each component per dimension].

perparameters such that the total validation loss is minimized. We train a new network with the updated network hyperparameters and repeat the Gaussian Process optimisation for approximately 30 iterations until a final optimised set of hyperparameters are chosen. We have found through random hyperparameter tuning that the size of the latent space does not have a significant effect on training performance as long as the latent space dimensionality is greater than the total number of source parameters inferred. Hence we have chosen a latent space size of $n_z = 10$ for the network.

Training procedure

Our cost function is composed of 3 probability distributions modelled by neural networks with well defined inputs and outputs where the mapping of those inputs to outputs is governed by the parameter sets θ_1, θ_2 and ϕ. These parameters are the weights and biases of 3 neural networks acting as (variational) encoder, decoder, and encoder respectively. To train such a network one must connect the inputs and outputs appropriately to compute the cost function H (Eq. 4) and back-propagate cost function derivatives to update the network parameters.

Training is performed via a series of steps illustrated

schematically in Fig. 1. A batch of data composed of pairs of time-series y and their corresponding true GW signal parameters x are passed as input and the following steps are applied to each element of the batch.

1. The encoder q_ϕ takes both the time-series y and the true parameters x defining the GW signal. It then encodes these instances into parameters μ_q defining an uncorrelated (diagonal covariance matrix) n_z-dimensional Gaussian distribution in the latent space.

2. The encoder r_{θ_1} is given only the time-series data y and encodes it into a set of variables μ_{r_1} defining a multi-component multivariate Gaussian mixture distribution in the latent space.

3. We then draw a sample from the distribution described by μ_q giving us a location z_q within the latent space.

4. This sample, along with its corresponding y data, are then passed as input to the decoder r_{θ_2}. This decoder outputs μ_{θ_2} comprising a set of parameters that define a distribution in the physical x space.

5. The first term of the loss function, the reconstruction loss (defined as L in Eq. 4), is then computed by evaluating the probability density defined by μ_{θ_2} at the true x training value (the average is then taken over the batch of input data).

6. The second loss component, the KL-divergence between the distributions $q_\phi(z|x,y)$ and $r_{\theta_1}(z|y)$ (described by the parameter sets μ_q and μ_{r_1}), is approximated as

$$\begin{aligned} &\mathrm{KL}\left[q_\phi(z|x_n,y_n)||r_{\theta_1}(z|y_n)\right] \\ &\quad\approx \log\left(\frac{q_\phi(z|x_n,y_n)}{r_{\theta_1}(z|y_n)}\right)\Bigg|_{z\sim q_\phi(z|x_n,y_n)} \end{aligned} \tag{13}$$

   where z is the sample drawn from $q_\phi(z|x_n,y_n)$ in the first training stage. We use this single-sample Monte-Carlo integration approximation since the KL-divergence between a single-component and a multi-component multivariate Gaussian distribution has no analytic solution (the average is then taken over the batch of input data).

7. The 2 loss components are then summed according to Eq. 4 and all trainable network parameters (defined by θ_1, θ_2, ϕ) are updated based on the derivative of the cost function with respect to these parameters.

A problematic aspect of training relates to the behaviour of the network during the initial stages of training. The network has a strong tendency to become trapped in local minima resulting in a decreasing cost component L (the reconstruction cost) but a non-evolving KL-divergence term that remains close to zero. To avoid this state we apply an annealing process in which the KL-divergence term is initially ignored but its contribution is then increased logarithmically from 0 to 1 between the iteration indices 10^4—10^5. This allows the q_ϕ encoder to learn the latent space representation of the data via the reconstruction cost before being required to try to best match its distribution to that modelled by the r_{θ_1} encoder. In parallel with the gradual introduction of the KL cost term, we also find that the stability of training is negatively affected by the complexity of our tailored output decoder likelihood functions. To resolve this we apply the same annealing procedure over the same iteration range in transitioning between unbound Gaussian likelihoods on all physical parameters to the tailored likelihoods.

As is standard practice in machine learning applications, the cost is computed over a batch of training samples and repeated for a pre-defined number of iterations. For our purposes, we found that $\sim 10^6$ training iterations, a batch size of 512 training samples and a learning rate of 10^{-4} was sufficient. We used a total of 10^7 training samples in order to adequately cover the BBH parameter space. We additionally ensure that an (effectively) infinite number of noise realizations are employed by making sure that every time a training sample is used it is given a unique noise realisation despite only having a finite number of waveforms.

Completion of training is determined by comparing output posteriors on test samples with those of `Bilby` iteratively during training. This comparison is done using standard figures of merit such as the p-p-plot KL-divergence (see Figs. 4 and 5). We also assess training completion based on whether the evolution of the cost function and its component parts (Fig. 3) have converged. We use a single Nvidia Tesla V100 GPUs with 16/32 Gb of RAM although consumer grade "gaming" GPU cards are equally fast for this application.

The testing procedure

After training has completed and we wish to use the network for inference we follow the procedure described in the right hand panel of Fig. 1. Given a new y data sample (not taken from the training set) we simply input this into the trained encoder r_{θ_1} from which we obtain a single value of μ_{r_1} describing a distribution (conditional on the data y) in the latent space. We then repeat the following steps:

1. We randomly draw a latent space sample z_{r_1} from the latent space distribution defined by μ_{r_1}.

2. The z_{r_1} sample and the corresponding original y data are fed as input to our pre-trained decoder network r_{θ_2}. The decoder network returns a set of

parameters μ_{r_2} which describe a multivariate distribution in the physical parameter space.

3. We then draw a random x realisation from that distribution.

A comprehensive representation in the form of samples drawn from the entire joint posterior distribution can then be obtained by simply repeating this procedure and hence sampling from our latent model $r_\theta(x|y)$ (see Eq. 3).

FIG. 4. One-dimensional p-p plots for each parameter and for each benchmark sampler and `VItamin`. The curves were constructed using the 256 test datasets and the dashed black diagonal line indicates the ideal result. The best and worst-case p-values associated with each sampling method are (0.972,0.211 `VItamin`), (0.832,0.043 `Dynesty`), (0.728,0.117 `ptemcee`), (0.840,0.189 `CPNest`), (0.489,0.002 `emcee`).

TABLE IV. Benchmark sampler configuration parameters. Values were chosen based on a combination of their recommended default parameters [18] and private communication with the `Bilby` development team.

sampler	parameters
`Dynesty` [15]	live-points = 5000 tolerance = 0.1
`ptemcee` [17]	walkers = 250 temperatures = 8 steps = 5000 burn = 4000
`CPNest` [14]	live-points = 5000 tolerance = 0.1
`emcee` [16]	walkers = 250 steps = 14000 burn= 4000

Additional tests

A standard test used within the GW parameter estimation community is the production of so-called p-p plots which we show for our analysis and the benchmark comparisons in Fig. 4. The plot is constructed by computing a cumulative probability for each 1-dimensional marginalised test posterior evaluated at the true simulation parameter value (the fraction of posterior samples $\leq$ the simulation value). We then plot the cumulative distribution of these values [5]. Curves consistent with the black dashed diagonal line indicate that the 1-dimensional Bayesian probability distributions are consistent with the frequentist interpretation - that the truth will lie within an interval containing $X\%$ of the posterior probability with a frequency of $X\%$ of the time. It is clear to see that results obtained using `VItamin` show deviations from the diagonal that are entirely consistent with those observed in all benchmark samplers. The p-value has also been calculated for each sampler and each parameter under the null-hypothesis that they are consistent with the diagonal. These results show that for at least 1 parameter, emcee shows inconsistency with the modal at the 1% level. `Dynesty` has a worst case that is consistent only at the 4% level. All other samplers (including `VItamin`) show consistency at $> 10\%$ in the worst case.

The KL-divergence has been used to define the cost function of the CVAE analysis, but in general it is used as measure of the similarity between distributions. In Fig. 5 we use this quantity to compare the output posterior estimates between samplers for the same input test data. To do this we run each independent sampler (including `VItamin`) on the same test data to produce samples from the corresponding posterior. We then compute the KL-divergence between the output distributions from each sampler with every other sampler [39]. For distributions that are identical, the KL-divergence should equal zero but since we are representing our posterior distributions using finite numbers of samples, identical distributions result in KL-divergence values $\mathcal{O}(X)$. In Fig. 5 we show the distributions of these KL-divergences for the 256 test GW samples. In each panel we plot the distribution of KL-divergences obtained when comparing one of the 4 benchmark samplers with all other benchmark samplers (excluding `VItamin`). We also plot the distribution of KL-divergences obtained when comparing the same sampler with `VItamin` alone. In all 4 cases the `VItamin` results show distributions completely consistent with the deviations observed between benchmark samplers.

* Corresponding author: h.gabbard.1@research.gla.ac.uk

[1] B. P. Abbott *et al.* (LIGO Scientific Collaboration and

FIG. 5. Distributions of KL-divergence values between posteriors produced by different samplers. In each panel we show the distribution of KL-divergences computed between a single benchmark sampler and every other benchmark sampler over all 256 GW test cases (grey). Also plotted in each panel are the corresponding KL-divergence distributions between the single benchmark sampler and the `VItamin` outputs (blue, green, purple, yellow).

Virgo Collaboration), Phys. Rev. X **6**, 041015 (2016).
[2] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Phys. Rev. Lett. **119**, 161101 (2017).
[3] B. P. Abbott *et al.*, Living Reviews in Relativity **21**, 3 (2018), arXiv:1304.0670 [gr-qc].
[4] A. C. Searle, P. J. Sutton, and M. Tinto, Classical and Quantum Gravity **26**, 155017 (2009), arXiv:0809.2809 [gr-qc].
[5] J. Veitch, V. Raymond, B. Farr, W. M. Farr, P. Graff, S. Vitale, B. Aylott, K. Blackburn, N. Christensen, M. Coughlin, W. D. Pozzo, F. Feroz, J. Gair, C.-J. Haster, V. Kalogera, T. Littenberg, I. Mandel, R. O'Shaughnessy, M. Pitkin, C. Rodriguez, C. Rver, T. Sidery, R. Smith, M. V. D. Sluys, A. Vecchio, W. Vousden, and L. Wade, Physical Review D (2014),

10.1103/PhysRevD.91.042003, arXiv:1409.7215.
[6] "Gracedb gravitational-wave candidate event database (ligo/virgo o3 public alerts)," `https://gracedb.ligo.org/superevents/public/O3/`, accessed: 2019-09-16.
[7] L. P. Singer and L. R. Price, Phys. Rev. D **93**, 024013 (2016), arXiv:1508.03634 [gr-qc].
[8] F. Tonolini, A. Lyons, P. Caramazza, D. Faccio, and R. Murray-Smith, "Variational inference for computational imaging inverse problems," (2019), to appear in JMLR, arXiv:1904.06264.
[9] A. Pagnoni, K. Liu, and S. Li, "Conditional variational autoencoder for neural machine translation," (2018), arXiv:1812.04405.
[10] D. George and E. Huerta, Physics Letters B **778**, 64 (2018).
[11] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).
[12] T. Gebhard, N. Kilbertus, G. Parascandolo, I. Harry, and B. Schölkopf, in *Workshop on Deep Learning for Physical Sciences (DLPS) at the 31st Conference on Neural Information Processing Systems (NIPS)* (2017).
[13] J. Skilling, Bayesian Anal. **1**, 833 (2006).
[14] J. Veitch, W. D. Pozzo, C. Messick, and M. Pitkin, (2017), 10.5281/zenodo.835874.
[15] J. S. Speagle, "dynesty: A dynamic nested sampling package for estimating Bayesian posteriors and evidences," (2019), arXiv:1904.02180.
[16] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, PASP **125**, 306 (2013), 1202.3665.
[17] W. Vousden, W. M. Farr, and I. Mandel, (2015), 10.1093/mnras/stv2422, arXiv:1501.05823.
[18] G. Ashton, M. Huebner, P. D. Lasky, C. Talbot, K. Ackley, S. Biscoveanu, Q. Chu, A. Divarkala, P. J. Easter, B. Goncharov, F. H. Vivanco, J. Harms, M. E. Lower, G. D. Meadors, D. Melchor, E. Payne, M. D. Pitkin, J. Powell, N. Sarin, R. J. E. Smith, and E. Thrane, Astrophysical Journal Supplement Series (2018), 10.3847/1538-4365/ab06fc, arXiv:1811.02042.
[19] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. K. Katsaggelos, S. L. Larson, T. K. Lee, C. Lintott, T. B. Littenberg, A. Lundgren, C. sterlund, J. R. Smith, L. Trouille, and V. Kalogera, Classical and Quantum Gravity **34**, 064003 (2017).
[20] M. Coughlin, P. Earle, J. Harms, S. Biscans, C. Buchanan, E. Coughlin, F. Donovan, J. Fee, H. Gabbard, M. Guy, N. Mukund, and M. Perry, Classical and Quantum Gravity **34**, 044004 (2017).
[21] P. Graff, F. Feroz, M. P. Hobson, and A. Lasenby, Monthly Notices of the Royal Astronomical Society **421**, 169 (2012), arXiv:1110.2997 [astro-ph.IM].
[22] A. J. K. Chua and M. Vallisneri, arXiv e-prints , arXiv:1909.05966 (2019), arXiv:1909.05966 [gr-qc].
[23] S. R. Green, C. Simpson, and J. Gair, arXiv e-prints , arXiv:2002.07656 (2020), arXiv:2002.07656 [astro-ph.IM].
[24] K. Cranmer, J. Brehmer, and G. Louppe, Proceedings of the National Academy of Sciences (2020), 10.1073/pnas.1912789117.
[25] K. Sohn, H. Lee, and X. Yan, in *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., 2015) pp. 3483–3491.
[26] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," (2015), arXiv:1512.00570.
[27] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug and play generative networks: Conditional iterative generation of images in latent space," (2016), arXiv:1612.00005.
[28] A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera, "Handling incomplete heterogeneous data using VAEs," (2018), arXiv:1807.03653.
[29] "Advanced LIGO sensitivity design curve," `https://dcc.ligo.org/LIGO-T1800044/public`, accessed: 2019-06-01.
[30] S. Khan, K. Chatziioannou, M. Hannam, and F. Ohme, "Phenomenological model for the gravitational-wave signal from precessing binary black holes with two-spin effects," (2018), arXiv:1809.10113.
[31] R. Smith, S. E. Field, K. Blackburn, C.-J. Haster, M. Pürrer, V. Raymond, and P. Schmidt, Physical Review D **94**, 044031 (2016), arXiv:1604.08253 [gr-qc].
[32] D. Wysocki, R. O'Shaughnessy, J. Lange, and Y.-L. L. Fang, Physical Review D **99**, 084026 (2019), arXiv:1902.04934 [astro-ph.IM].
[33] C. Talbot, R. Smith, E. Thrane, and G. B. Poole, Physical Review D **100**, 043030 (2019), arXiv:1904.02863 [astro-ph.IM].
[34] B. P. Abbott *et al.*, Astrophysical Journal Letters **892**, L3 (2020), arXiv:2001.01761 [astro-ph.HE].
[35] T. B. Littenberg and N. J. Cornish, Phys. Rev. D **91**, 084034 (2015), arXiv:1410.3852 [gr-qc].
[36] P. Gallinari, Y. LeCun, S. Thiria, and F. F. Soulie, in *Proceedings of COGNITIVA 87, Paris, La Villette, May 1987* (Cesta-Afcet, 1987).
[37] D. J. Siria, R. Sanou, J. Mitton, E. P. Mwanga, A. Niang, I. Sare, P. C. Johnson, G. Foster, A. M. Belem, K. Wynne, R. Murray-Smith, H. M. Ferguson, M. González-Jiménez, S. A. Babayan, A. Diabaté, F. O. Okumu, and F. Baldini, bioRxiv (2020), 10.1101/2020.06.11.144253.
[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).
[39] Q. Wang, S. R. Kulkarni, and S. Verdu, IEEE Transactions on Information Theory **55**, 2392 (2009).