Gravitational Wave modelling with Machine Learning

SB: [Alt.]Machine Learning Gravitational Waves from Binary Black Hole Mergers

Stefano Schmidt,[1] Matteo Breschi,[1] Rossella Gamba,[1] Giulia Pagano,[1] Piero Rettegno,[1] Sebastiano Bernuzzi,[1] Alessandro Nagar,[1] and Walter Del Pozzo[1]

[1]*Dipartimento di Fisica "Enrico Fermi", Università di Pisa, and INFN Sezione di Pisa, Pisa I-56127, Italy*

We apply machine learning methods to build a time-domain model for gravitational waveforms from binary black hole mergers called `mlgw`. The dimensionality of the problem is handled by representing the waveform's amplitude and phase using a principal component analysis. We train `mlgw` on about $\mathcal{O}(10^3)$ `TEOBResumS` effective-one-body waveforms with mass ratios $q \in [1, 20]$ and aligned dimensionless spins $\chi \in [-0.80, 0.95]$. The resulting model is faithful to the training set at the $\sim 10^{-3}$ level (averaged on the parameter space), while speeding up the single waveform generation of a factor 10-50 (depending on the binary mass and initial frequency.) Furthermore, `mlgw` provides a closed form expression for the waveform and its gradient with respect to the orbital parameters; such an information might be for future improvements in GW data analysis.

As demonstration of the capabilities of `mlgw` to perform a full parameter estimation, we re-analyze the public data from the first GW transient catalog (GWTC-1) to find consistent results at a fraction of the cost. Since the generation time does not depend on the length of the signal, our model is particularly suitable for the analysis of long signals and/or and in view of third-generation detectors. Future applications include the analysis of waveform systematics and model selection in parameter estimation.

CONTENTS

I. INTRODUCTION

The detection of Gravitational Waves (GW) from compact binary coalescences (CBC) has been possible thanks to the joint effort of a number of different fields of expertise, all joining together to achieve the sophisticated detection process. GW data analysis concerns the detection of a GW signal hidden in the raw detector output (*matched filtering*) and subsequently the inference of its physical properties (*parameter estimation*). In order to accomplish its goal, GW data analysis relies on the availability of waveform (WF) templates to compare with the detector output. To accurately explore the posterior distribution for the parameters defining a CBC, state-of-the-art parameter estimation (PE) algorithms [?] [?] can require the generation of as many as 10^7 waveform templates. It is therefore paramount for the waveform generation to be as fast as possible. At the same time, because of the extreme sensitivity to phase differences in the likelihood function, the templates must retain a high degree of accuracy to avoid biases in the posterior exploration. Many efforts have been devoted to numerically solve Einstein equations for two coalescing objects and to predict the gravitational radiation emitted []. As solving the full equations is still extremely computationally challenging, the LIGO-Virgo Collaboration relies on approximate analytical models. These can be broadly categorized in two families; (i) the effective-one-body(EOB) waveform models [? ?]; (ii) the phenomenological models [? ? ?]. SB: [would add NRsurrogates] EOB models are the most complete family of analytical models available. They compute the GW signal by solving Hamilton's equations and accurately predict the GW signal up to late plunge. The merger and ringdown parts of the signal are then joint to the inspiral signal using information obtained from numerical relativity. Because of the numerical integrations involved, they tend to be accurate, but slow to generate, see however [?] for a more efficient approach to obtain the dynamics.

The phenomenological waveforms are based on the post-Newtonian formalism and then calibrated to EOB waveforms and numerical relativity. They tend to be faster than EOB models, but not as accurate. Many efforts have been devoted to the task of speeding up the generation of GW signals from EOB families. For example, one lead to the development of *surrogate models*.

Surrogate models are constructed starting from some decomposition in the waveform space followed by efficient interpolation to avoid any numerical integration [? ? ? ? ?]. Being fast to generate, they are routinely employed in GW data analysis.

A Machine Learning model is a promising alternative to accelerate the waveforms generation of state-of-the-art models. Machine Learning (ML) is a branch of statistics that is devoted to reproduce patterns read from data. A ML algorithm needs very little human input and, by automatically solving an optimization problem, it is able to choose the best performing element among a large class of parametric models for the solution. This is the so-called training procedure. The ML flexibility in modeling data and reproducing trends is appealing: with a proper model choice and with an appropriate training procedure, we can hope to have a reliable, fast to execute generator of GW waveforms, while retaining the accuracy necessary for robust parameter estimation. ML procedures have been applied so far to Numerical Relativity waveform data [? ? ? ? ?]. SB: [check carefully refs, I think there are more][also note the Gabbard paper uses EOB not NR, I am not sure of the sentence ''ML is applied so far to NR''] The construction of a NR-trained ML waveform model is currently missing except for a very restricted region of the parameter space (nonspinning etc.). The main problem behind this is because of the absence of a large set of NR simulations ($\simeq 10^3$) that would be needed to train such an algorithm.

By contrast, an EOB model, informed by NR simulations, does not have such drawback since it is designed to generate waveforms all over the BBH parameter space. One has indeed to remember that, although NR-informed EOB models are *checked* only on the part of the parameter space covered by NR simulations, they are also typically able to robustly *extrapolate* waveforms outside this NR-information domain (e.g. large mass ratios and spins). SB: [more than extrapolating I would say something technical like] well-defined for any adiabatic inspirals from PN knowledge, informed in test-mass regime, defined for arbitrary length, etc... This is, for example, the case of the most recent avatar of the spin-aligned `TEOBResumS` [?] model, that incorporates subdominant waveform modes, `TEOBResumS_SM` [? ?]. This model used several hundreds of the available SXS simulations to inform a highly accurate description of the postmerger-ringdown phase [?], but only around 40 datasets to improve the behavior of the analytical EOB dynamics, suitably modifying the analytical orbital and spin-orbit interaction. Thus we use `TEOBResumS_SM` to train our machine learning model for BBH coalescence. For simplicity, we focus here only on the dominant $\ell = m = 2$ quadrupole waveform. We demonstrate that such ML-based model can generate GW signals significantly faster than the training model, matching the performances of a Reduced Order Modelling (ROM) [? ? ?]. At the same time it keeps the same accuracy of the underlying training model[1].

As a relevant physical application, we use the our ML model `mlgw` to provide a completely new, and independent, analysis of the 10 BBHs coalescence events collected in the O1/O2 LIGO-Virgo observing runs [?]. The physical parameters we find are compatible with previous analyses obtained with radically different models [?]. On top of the specific application discussed here, our ML waveform model could also be used directly to speed up GW data analysis. Furthermore, since the time required to generate a WF does not depend on the signal time length but only only on the number of grid points which the WF is evaluated at, the applicability of our approach goes far beyond the LIGO/Virgo physics scenario. In particular, we think about the forthcoming Einstein Telescope, that will be sensitive to very long stellar-mass inspirals from 5Hz, or to extreme mass ratio inspiral as LISA sources. In these context the problem of WF fast generation will be more pressing and our approach, provided a suitable waveform model for training, might be essential for detection and parameter estimation.

The paper is organized as follows: in section II we briefly set the notation and the core of the ML problem we solve; in section III we describe our model in details; section IV we validate the model and assess its accuracy and speed of execution; section V holds our analysis of the GWTC-1 transient catalog; finally in VI we will report some final remarks and future prospects of our work.

II. CONVENTIONS SETUP

A binary black hole is parametrized by a vector $\boldsymbol{\vartheta} = (m_1, m_2, \mathbf{s}_1, \mathbf{s}_2)$, where m_i are the BHs masses and $\mathbf{s}_i = \frac{\mathbf{S}_i}{m_i^2} < 1$ are the *dimensionless* spin. We call them the *orbital parameters*. Let the wave direction of propagation be identified with the spherical coordinates (d_L, ι, φ_0), where d_L is the luminosity distance, ι the polar angle and φ_0 the azimuthal angle. The polar angle ι is also called inclination and it is measured with respect to the normal to the orbital plane. A GW is parametrized as:

$$\begin{aligned} h(t, d_L, \iota, \varphi_0; \boldsymbol{\vartheta}) &= h_+ + i h_\times \\ &= \sum_{l=2}^{\infty} \sum_{m=-l}^{l} {}^{-2}Y_{lm}(\iota, \varphi_0) \cdot H_{lm}(t; \boldsymbol{\vartheta}) \end{aligned} \tag{1}$$

where ${}^{-2}Y_{lm}(\iota, \varphi_0)$ are the spin-2 spherical harmonics. We call $(m_1, m_2, \mathbf{s}_1, \mathbf{s}_2, d_L, \iota, \varphi_0)$ *physical parameters* and they fully express the source proprieties as well as its orientation and position. In what follows, we will focus

[1] Let us note, in passing, that the ML framework is completely general and it works for any EOB model; indeed a successful ML model was trained using also the `SEOBNRv2` waveform model [?].

on the case in which spins $\mathbf{s}_1$ and $\mathbf{s}_2$ are *aligned* with the orbital angular momentum.

In this work, we concentrate on the single complex quantity H_{22}. Since the dependence on the two angles and distance are known, we fix their value to $\iota, \varphi_0 = 0$ and $d_L = 1\,\text{Mpc}$ and we only work with waveforms $h_{FIT}(t;\boldsymbol{\vartheta}) = h(t; d_L = 1\,\text{Mpc}, \iota = 0, \varphi_0 = 0, \boldsymbol{\vartheta})$:

$$h_{FIT}(t;\boldsymbol{\vartheta}) \equiv {}^{-2}Y_{22}\,(0,0)\cdot H_{22}(t;\boldsymbol{\vartheta}) = 4\cdot\sqrt{\frac{5}{64\pi}}H_{22}(t;\boldsymbol{\vartheta})\;. \tag{2}$$

Finally, we express h_{FIT} in terms of its amplitude and phase [2]:

$$h_{FIT}(t;\boldsymbol{\vartheta}) = A(t;\boldsymbol{\vartheta})e^{i\phi(t;\boldsymbol{\vartheta})}\;. \tag{3}$$

We may also write $f_{\boldsymbol{\vartheta}}(t)$ to denote a function $f(t;\boldsymbol{\vartheta})$ of time with parametric dependence on $\boldsymbol{\vartheta}$. In what follows, f stands for any of the functions $A_{\tilde{\boldsymbol{\vartheta}}}(t)$ and $\phi_{\tilde{\boldsymbol{\vartheta}}}(t)$.

With this definition, the full waveform can be expressed as:

$$\begin{aligned} &h(t, d_L, \iota, \varphi_0; \boldsymbol{\vartheta}) = \\ &= \frac{1\,\text{Mpc}}{d_L}\cdot\left\{\frac{1+\cos^2\iota}{2}\cdot A_{\boldsymbol{\vartheta}}(t)\cos[\phi_{\boldsymbol{\vartheta}}(t) + 2\varphi_0]\right. \\ &\left. + i\cdot\cos\iota\cdot A_{\boldsymbol{\vartheta}}(t)\sin[\phi_{\boldsymbol{\vartheta}}(t) + 2\varphi_0]\right\} \end{aligned} \tag{4}$$

where we split the real and the imaginary part of h and we used the relation ${}^{-2}Y_{2\pm2}\,(\iota,\varphi_0) = (1\pm\cos\iota)^2 e^{\pm i2\varphi_0}$. We choose the convention that $\phi_{\boldsymbol{\vartheta}} = 0$ when the amplitude $A_{\boldsymbol{\vartheta}}$ has a maximum [3].

`AN: All text above should be simplified!!`

III. THE MODEL `SB: [HOW ABOUT:] MLGW`

The goal of the present work is to provide an accurate Machine Learning model which outputs the functions $A(t;\boldsymbol{\vartheta})$ and $\phi(t;\boldsymbol{\vartheta})$ (eq. (3)), as generated by the state-of-the-art time domain WF models. Since BBH waveforms trivially scales with the binary mass M, we only need to consider the variables $\tilde{\boldsymbol{\vartheta}} = (q, s_1, s_2)$, where $q = m_1/m_2 \geq 1$, to describe the waveform's multipoles. More formally, we seek a ML model that reliably reproduces the following map:

$$(q, s_1, s_2) \longmapsto A_{(q,s_1,s_2)}(t) \tag{5}$$
$$(q, s_1, s_2) \longmapsto \phi_{(q,s_1,s_2)}(t) \tag{6}$$

[2] Note we adopt a nonstandard sign convention for the phase
[3] Indeed, a constant translation of $\phi_{\boldsymbol{\vartheta}}$ can be absorbed in the definition of φ_0 and does not affect the physics.

In the context of ML, our task reduces to performing two regressions[4] from $\tilde{\boldsymbol{\vartheta}}$ to the amplitude and phase of WF. In order to be able to perform each regression, several steps are required.

(A) *Setting a time grid.* Each WF must be represented on a discrete time grid, which allows for efficient and reliable reconstruction on an arbitrary, user-given, grid. After this operation, the functions $A(t)$ and $\phi(t)$ are represented as vectors[5].

(B) *Creating a dataset of WFs.* A large number of WFs must be generated on the the chosen time grid for a different number of orbital paramter (q, s_1, s_2). This will be the training set for the ML model.

(C) *Reducing the dimensionality of a WF.* In order to make the regression feasible, we build a low dimensional representation of the WF. This operation must be invertible: once a low dimensional representation is given, one should be able to reconstruct the higher dimensional WF.

(D) *Learning a regression.* We train a model to perform the regression from (q, s_1, s_2) to the low dimensional representation of the WF.

We discuss these points in detail in what follows.

A. The time grid

Each function $f(t)$ to fit (i.e. amplitude and phase) must be represented by its values $\mathbf{f} \in \mathbb{R}^D$ on a discrete grid of D points $\mathbf{t} \in \mathbb{R}^D$. It is convenient to work in a grid of (dimensionless) reduced time $\boldsymbol{\tau} = \frac{\mathbf{t}}{M}$. The time grid is chosen with the convention that at $\tau = 0$ the function $A(t;\boldsymbol{\vartheta})$ (i.e. the amplitude of the 22 mode) has a peak. Once a time grid is set, the vector $\mathbf{f}$ is defined as follows:

$$\mathbf{f}(\tilde{\boldsymbol{\vartheta}})_i = f_{\tilde{\boldsymbol{\vartheta}}}(\boldsymbol{\tau}_i) \qquad i = 1, \ldots D \tag{7}$$

The value of f at an arbitrary time must be found by interpolation and this introduces an error in the reconstructed value. To make the interpolation effective, we introduce a grid adapted to the function's variation. Clearly an equally spaced grid over times is not the best

[4] A regression is a statistical method to infer the relationship between a set of "independent variables" and a set of "dependent variable". A model consist in a functional form for such relation, usually with many free parameters to be specified. By looking at the data, one should be able to make a proper choice for their value.
[5] In ML jargon, this procedure is called preprocessing and aims to create a standard representation for all the data available (in our case the WFs).

choice since the amplitude has a very narrow peak at $\tau = 0$[6]. A good solution is to build the τ grid $\boldsymbol{\tau}$ as:

$$\boldsymbol{\tau}_i = \operatorname{sign}\tilde{\boldsymbol{\tau}}_i \cdot (|\tilde{\boldsymbol{\tau}}_i|)^{\frac{1}{\alpha}} \qquad i = 1, \dots D \tag{8}$$

where $\tilde{\boldsymbol{\tau}}_i$ are D equally spaced points in the range of interest and we call α *distortion parameter*. This choice ensures that more points are accumulated around the peak of amplitude.

The length of the time grid determines the maximum lenght of the WFs that the model can generate. Let us define $\tau_{min} = -\boldsymbol{\tau}_0 > 0$ the starting point of the grid; thus each WF start at a time $\tau_{min} \cdot M$ before the merger. Note that τ_{min} is an important hyperparameter, set by the user, which strongly impacts on the model applicability.

The minimum frequency in the signal as a function of M, q and τ_{min} is given approximately[7] by:

$$f_{min} = 151\,\text{Hz} \left(\frac{(1+q)^2}{q}\right)^{\frac{3}{8}} \left(\frac{M_\odot}{M}\right) \left(\frac{1\,\frac{\text{s}}{\text{M}_\odot}}{\tau_{min}}\right)^{\frac{3}{8}} \tag{9}$$

B. Dataset creation

As in any ML method, we must create a dataset before training any model. In our case, the dataset consist in a matrix $X \in \mathbf{Mat}(N, 3+2D)$ of N waveform, which has the following form:

$$X_{i:} = [q, s_1, s_2, A^T_{\tilde{\boldsymbol{\vartheta}}}, \boldsymbol{\phi}^T_{\tilde{\boldsymbol{\vartheta}}}] \tag{10}$$

where $X_{i:}$ denotes the i-th row of the dataset matrix.

The dataset is filled with random values of parameters $\tilde{\boldsymbol{\vartheta}}_i \sim \text{Unif}(\mathcal{P})$ [8]. To generate the training waves the `TEOBResumS` model is used. Waves in eq. (2) are generated with a standard total mass $M = 20\,\text{M}_\odot$.`SB: [strange sentence (generating waves in an equation...) and we just said the total mass is not important ... remove?]` The output of the training model must be interpolated to the chosen time grid.

It is important to ensure that all waves have zero phase at a constant time point $\bar{t}$: this is crucial to obtain a continuous dependence of the phase components on the orbital parameters. As model performances are not seen to depend on the choice of $\bar{t}$, we arbitrarly set $\bar{t} = 0$.

The range $\mathcal{P}$ of masses and spins covered by the model, as well as the starting point of the grid τ_{min}, can be freely choose by the user, depending on their needs.

[6] As the phase has a rather regular behavior, it is not important to choose the right grid. For this reason a single grid for amplitude and phase, tuned on the behavior of amplitude, is used.

[7] The expression is approximate because it is obtained within a Newtonian framework and does not consider spin effects. Nevertheless, it gives an useful estimation of the range of the applicability of the model.

[8] We denote by Unif($\mathcal{P}$) a uniform probability distribution on the set $\mathcal{P}$.

C. Dimensionality reduction

Once we are able to represent waveforms, a regressions $\tilde{\boldsymbol{\vartheta}} \longmapsto \boldsymbol{A}_{\tilde{\boldsymbol{\vartheta}}}, \boldsymbol{\phi}_{\tilde{\boldsymbol{\vartheta}}}$ is unfeasible, as the dimension of the target space is too large. Luckily, the elements of $\boldsymbol{A}, \boldsymbol{\phi}$ are strongly correlated with each other: the independent amount of information, required to fully reconstruct the wave, can be stored in a low dimensional vector. A number of ML techniques to perform such a task are available. Among them, Principal Component Analysis (PCA) [**?** , ch. 12] was found to be particularly effective.

The basic idea behind PCA is to seek a *linear relation* between high dimensional and low dimensional data: high dimensional data ($\in \mathbb{R}^D$) are projected onto a K dimensional subspace, by means of an orthogonal projection. A theorem [**?** , sec. 12.2.1] guarantees that, for zero mean data, the generators of subspace are the (orthonormal) first K eigenvectors of the empirical covariance matrix $\Sigma \in \mathbf{Mat}(D, D)$. The eigenvectors are also called Principal Components (PCs) of the data. Thus, the projection matrix $H \in \mathbf{Mat}(K, D)$ holds in each row the PCs and each high-dimensional point can be effectively expressed as a linear combination of the K PCs[9].

A PCA model is trained with the dataset eq. (10): it represents an (approximate) bijective map between the high dimensional WF $\mathbf{f} = \boldsymbol{A}_{\tilde{\boldsymbol{\vartheta}}}, \boldsymbol{\phi}_{\tilde{\boldsymbol{\vartheta}}} \in \mathbb{R}^D$ and the low-dimensional representation $\mathbf{g} = \mathbf{g}_A, \mathbf{g}_\phi \in \mathbb{R}^K$. The relation takes the following form:

$$\mathbf{g} = H(\mathbf{f} - \boldsymbol{\mu}) \tag{11}$$

$$\mathbf{f} = H^T\mathbf{g} + \boldsymbol{\mu} \tag{12}$$

where $\boldsymbol{\mu}$ is the empirical mean vector $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{f}_i \in \mathbb{R}^D$ and the matrix H is computed from the empirical covariance $\Sigma = \frac{1}{N}\sum_{i=1}^N (\mathbf{f}_i - \boldsymbol{\mu})(\mathbf{f} - \boldsymbol{\mu})^T_i$.

D. Regression

Once a dimensional reduction (and reconstruction) scheme is available, we want to perform the regression

$$\tilde{\boldsymbol{\vartheta}} \longmapsto \boldsymbol{g}(\tilde{\boldsymbol{\vartheta}}) \tag{13}$$

A number of ML models are available for this purpose. The model Mixture of Experts (MoE) [**?**] [**?** , ch. 11] is found to be a good compromise between simplicity and flexibility.

MoE performs the following 1D regression:

$$y(\mathbf{x}) = \sum_{l=1}^{L} (W^T\mathbf{x})_l \cdot \mathcal{S}(V^T\mathbf{x})_l \tag{14}$$

[9] For this reason, PCA can also be seen as a perturbative expansion of a high dimensional observation. A more realiable reconstruction can be achieved by adding more and more PCs, each of which is less important than its previous.

where $\mathcal{S}$ is the *softmax function*:

$$\mathcal{S}(V^T\mathbf{x})_l = \frac{e^{(V^T\mathbf{x})_l}}{\sum_{l'=1}^{L} e^{(V^T\mathbf{x})_{l'}}} \tag{15}$$

and $\mathbf{x} \in \mathbb{R}^{\tilde{M}}$ and $V, W \in \mathbf{Mat}(\tilde{M}, L)$. The meaning of eq. (14) is clear: the output is a weighted combination of L linear regressions $(W^T\mathbf{x})_l$ (called *experts*); each expert performs a reliable regression in a small region of the space. The softmax function (in this context also called *gating function*) switches on the expert contributions whenever this is required. MoE is usually fitted with the Expectation Maximization (EM) algorithm, which iteratively sets the W and V by refining a lower bound to the log-likelihood of the model.

Linear regression is a very simple model, often inadequate to model a complex relation. A simple trick to improve its performance is called *basis functions expansion*. It consist in the replacement:

$$\mathbf{x} \longrightarrow \boldsymbol{\xi}(\mathbf{x}) = [\xi_1(\mathbf{x}), \ldots, \xi_M(\mathbf{x})]^T \tag{16}$$

Thus, each expert becomes a non linear regression of the input $\mathbf{x}$. A careful choice of basis functions can really make a difference in fit performances and it must be done at validation time, by comparing performances of different models.

The user must choose the number L of experts and the basis functions features $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) \in \mathbb{R}^M$ to use. Including in the ξ_i every monomial up to 3/4th order in the three variables $(\log q, s_1, s_2)$ seems a good working choice for our model [10].

As MoE model deals with single dimensional outputs, a single independent regression must be performed for each component g_k of $\mathbf{g} \in \mathbb{R}^K$ [11]. In general, a regression will be a collection of MoE weights $\{W^{(k)}, V^{(k)} \in \mathbf{Mat}(M, L_k)\}_{k=0}^{K}$, where index k labels different regressions for each PC.

E. Summary

The model has the following explicit form:

$$\begin{aligned}
&\text{model} : \mathcal{P} \subset \mathbb{R}^3 \to \mathbb{R}^K \to \mathbb{R}^D \\
&\tilde{\boldsymbol{\vartheta}} \longmapsto \mathbf{g}(\tilde{\boldsymbol{\vartheta}}) = \begin{pmatrix} \sum_{l=1}^{L_1} (W^{(1)\,T}\boldsymbol{\xi})_l \cdot \mathcal{S}(V^{(1)\,T}\boldsymbol{\xi})_l \\ \vdots \\ \sum_{l=1}^{L_K} (W^{(K)\,T}\boldsymbol{\xi})_l \cdot \mathcal{S}(V^{(K)\,T}\boldsymbol{\xi})_l \end{pmatrix} \\
&\qquad \longmapsto \mathbf{f}(\tilde{\boldsymbol{\vartheta}}) = H^T \mathbf{g}(\tilde{\boldsymbol{\vartheta}}) + \boldsymbol{\mu}
\end{aligned} \tag{17}$$

where $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) \in \mathbb{R}^M$ are the chosen basis function for the regression and $\mathcal{S}(\cdot)_k$ is the *softmax* function eq (15). Two relations of the same type must be fitted, one for the amplitude, the other for the phase.

Once weights are set properly, the expression provides an estimation for the waveform h_{FIT} in (2). The fitted expression for h_{FIT} is evaluated at constant mass $M = 20\,\mathrm{M}_\odot$; the dependence on total mass is inserted analytically and the dependence on (d_L, ι, φ_0) is computed with eq. (4). We are thus able to obtain a complex waveform $h(t; m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0)$ which reproduces closely a waveform from the training model. The model can extrapolate outside the range of train orbital parameters, without guarantee of reliable results.

Note that equation 17 can be used to compute a closed form expression for the gradients of the waveform with respect to the orbital parameters. Such calculations are included in the released version of `mlgw`.

IV. MODEL PERFORMANCE

We now discuss some tests on our model. We first study how its performance depends on the choice of hyperparameters. Furthermore, we assess the model accuracy and its limitations. Finally, we measure the speed up provided by our model as compared with training `TEOBResumS` model. As it is common, we measure the similarity between two waves by means of the *optimal mismatch*:

$$\bar{\mathcal{F}}[h_1, h_2] \equiv 1 - \frac{\langle h_1, h_2 \rangle}{\sqrt{\langle h_1, h_1 \rangle \langle h_2, h_2 \rangle}} \tag{18}$$

where, as usual, we defined the *Wiener product* as:

$$\langle h_1, h_2 \rangle = 4 \int_0^\infty \mathrm{d}f \, \frac{h_1^*(f) h_2(f)}{S_n(f)} \tag{19}$$

In the equation above, $S_n(f)$ is the detector's noise curve and the $*$ denotes complex conjugate. In what follows, we always use a flat noise curve (i.e. constant power spectral density for the detector noise).

A. Validation

Wherever relevant, we will employ a dataset with 5800 waveforms generated in the domain $\mathcal{P} = [1, 20] \times [-0.8, 0.95] \times [-0.8, 0.95]$, with $\tau_{min} = 1.0\,\mathrm{s}/\mathrm{M}_\odot$. The dataset was generated with TEOBResumS model [].

a. Dataset generation parameters We first evaluate the impact of number of grid points N_{grid} and distortion parameter α. Let $\mathbf{f}_{N_{grid},\alpha}$ the wave stored in a dataset where τ_{min} and $\mathcal{P}$ are fixed as above. We compare it with the output of the EOB model $\mathbf{f}_{EOB}$. We then vary N_{grid} and α and report the resulting mismatch $\mathcal{F}[\mathbf{f}_{EOB}, \mathbf{f}_{N_{grid},\alpha}]$ in figure 1.

[10] The choice of working with variable $\log q$ rather than q gives much better validation results. Heuristically, using $\log q$ prevents the values of the data features to vary too much within the range of interest, yielding more stable numerical performance.

[11] This is not a great limitation, because, due to orthogonality of PCs, each g_j is independent from the other: we do not miss correlation among different regressions.

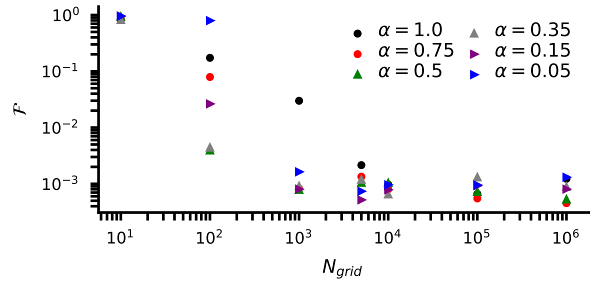

FIG. 1. Mismatch between waves $\mathbf{f}_{N_{grid},\alpha}$ and raw waves from EOB model, as a function of time grid size N_{grid}. Each series refers to a different values of α. Mismatch is computed on 10 test waves. textwidth: 3.40457in

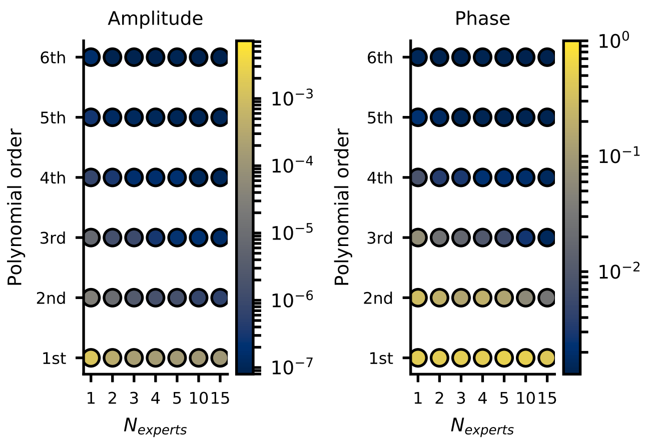

FIG. 2. Validation results for fit of MoE model. Each point corresponds to a MoE regressions for the amplitude (left) and phase (right), with a different values of expert number N_{exp} and order of polynomial basis function. The amplitude and phase are represented with 5 and 4 PCs respectively. In the colorbar, we represent the mismatch on test waves: it is obtained by reconstructing test waves with fitted amplitude (phase) and test phase (amplitude).

As expected, we note that, by increasing the number of grid points, the mismatch decreases. Furthermore, using more than $\sim 10^3$ grid points, does not bring any improvement to mismatch. In this case, the result is dominated by numerical errors in the interpolations and it provides a lower-bound for the performances of the fit. A careful choice of α provides a remarkable improvement when N_{grid} is small. For a high number of grid points, different values of α yield almost equivalent results. A good setting for dataset hyperparameters might be: $N_{grid} \simeq 2/3 \cdot 10^3$ and $\alpha \simeq 0.3/0.5$.

b. MoE parameters We only focus on setting the number of experts N_{exp} for each component model and the basis functions $\xi_i(\tilde{\boldsymbol{\vartheta}})$ to use in the regression. Other parameters, related to the details of the training procedure, will not be considered here.

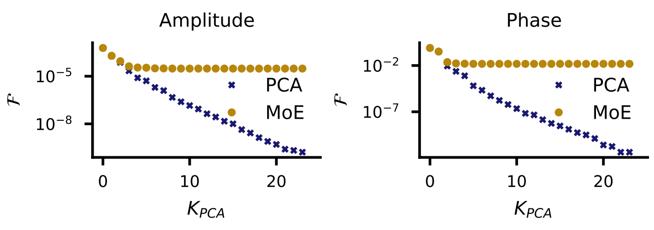

FIG. 3. Test mismatch as a function of the number of PCs used in the low dimensional representation. Label "PCA" refers to waves reconstructed with PCA only; points with label "MoE" are reconstructed after a MoE regression. Data refers to amplitude (left panel) and phase (right panel). MoE model is chosen to be the optimal one with 4 experts and a fourth order polynomial.

In figure 2 we present our results. We fitted a model for amplitude (or phase) for different configurations of expert number N_{exp} and polynomial basis function. By label "n-th order", we mean that in the basis function expansion, every monomial up to n-th order is used. We report with a colorbar the value of the mismatch F between test and reconstructed WFs. The MoE models for each component share the same number of experts N_{exp}. The test mismatch for the fitted amplitude (phase) is computed by using the test phase (amplitude) in the reconstructed wave.

As a general trend, fit performance improves whenever the model complexity (i.e. number of fittable parameters) increases. In general, we note that adding more features is more effective than employing the number of experts. However, the model performance does not improve indefinitely: as we see in figure 2, many "complex" models show similar performance, regardless their complexity. A model with 4 experts and 4th order polynomial regression is the "simpler" of such models and thus it should be deemed as the best choice.

c. Choosing the number of PCs Of course, the accuracy of the reconstruction of the low dimensional representation depends on the number K of principal components considered: the more PCs are used, the best accuracy can be achieved. However in practice, due to error in the MoE regression, one cannot reduce the reconstruction mismatch arbitrarily[12] and one should choose the number of PCs, while checking MoE performance.

In figure 3 we report a numerical study of this. We plot the reconstruction mismatch as a function of the number of PCs considered. We consider separetely the amplitude and the phase. In one series, we reconstruct the wave using true values of PCs: the mismatch is a measure of PCA accuracy. In the other, we reconstruct a wave using

[12] Indeed, at high PC order the relations to fit become noisy and the regression becomes less accurate, eventually washing out any improvement brought by a higher number of PCs.

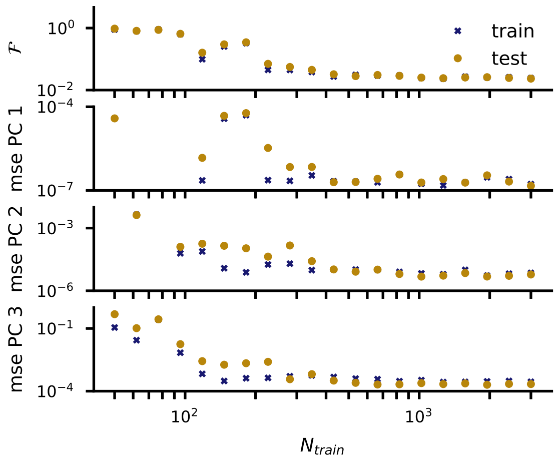

FIG. 4. Train and test error for MoE fit of 4 PCs of phase, as a function of the number of training points. We report train and test reconstruction mismatch (top) and mse for the first 3 PCs (below). MoE model employs 4 experts and a fourth order polynomial for a basis function expansion. Test mismatch are obtained using test amplitude to reconstruct the waveform; this is not a great limitation as any error in phase reconstruction dominates the overall mismatch.

values for PCs as guessed by MoE regression: this is a measure of accuracy of both PCA and regression. For the first two PCs, the regression is accurate enough for reproducing the PCA accuracy. On the other hand, any regression beyond the 4th PCA component does not give any improvement to the MoE mismatch: the noise in the relation of high order PCs is too high for a regression to be performed.

In the PCA, we include every PC which yields improvement in MoE mismatch. For our model, $K = 5(4)$ is a good choice for amplitude (phase). Of course, this strongly depends on the regression model: the more precise the model is, the more PCs can be included. However, any model cannot increase its accuracy indefinitely. Every training model has an intrinsic noise level, due to numerical error and to the approximations in the physical model.

d. Choosing the number of training points The choice of the number of training points N_{train} must trade between accuracy and speed of execution. Too many training points will make the training slow, while too few training points will yield a poor model, which does not generalize the data (underfitting). In the choice of number of training points, the comparison between train and test error will provide important information on how the model is able to generalize the trend.

In figure 4 we report train and test value of mismatch and mse of first 3 PCs as a function of the number of training points. Data refers to a MoE model fitted for 4 PCs of the phase dataset. The models has $N_{exp} = 4$ and performs basis function expansion with a 4th degree polynomial.

As N_{train} increases, we see a steady decrease of the

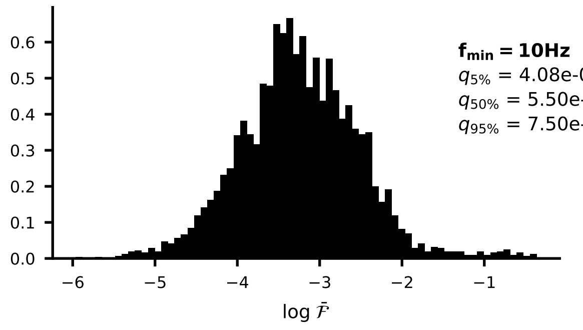

FIG. 5. Histogram for the logarithm of mismatch values, computed on $N = 4000$ test waveforms. Each WF is generated with random orbital parameters $(m_1, m_2, s_1, s_2, d_L, \iota, \phi_0)$ and with a starting frequency of 10 Hz. We report the median value $q_{50\%}$ as well as the positions $q_{5\%}$ and $q_{95\%}$ of the 5th and 95th percentile. The 95% of the test waves have a mismatch lower than $7.5 \cdot 10^{-3}$.

errors, until a plateau is reached. Since for a reasonably high number of training points ($N_{train} \gtrapprox 50$) train and test error are close to each other, we note that overfitting is not a problem. For $N_{train} \gtrapprox 800$, the trend stabilises and increasing training points does not affect much model performance. In the present model, setting $N_{train} \simeq 3000$ is a good choice [13].

B. Accuracy

We compute mismatch value on a number of WFs with random values of the physical parameters (i.e. $m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0$) for each wave. We report our results in the histogram in figure 5. We report a median value of the distribution $\mathcal{F}_m = 5.5 \cdot 10^{-4}$. Such results are similar to the discrepancies between state-of-the-art models.`SB: REFS`

To understand better model performances, it is interesting to display the accuracy as a function of the orbital parameters $\boldsymbol{\vartheta} = (q, M, s_1, s_2)$. We generate waves for randomly chosen values of $\vartheta = (q, M, s_1, s_2)$ and, for each wave, we measure test mismatch $\mathcal{F}$ and mse on the reconstruction of the first PC for the phase. The latter is useful to test the accuracy of the fit before wave reconstruction. The results are reported in the contour plots in figure 6. `SS: Probably the comments below are not really interesting...`

By looking at the top line of 6, we note that the mse does not depend on M. This was to be expected because the total mass dependence is inserted analytically within

[13] As compared with standard neural networks, which routinely employ $O(10^5)$ points datasets, this is an incredibly low amount of data. This is due to the fact that MoE is a simple model with a few number of parameters: few data are enough for learning a reliable relation.

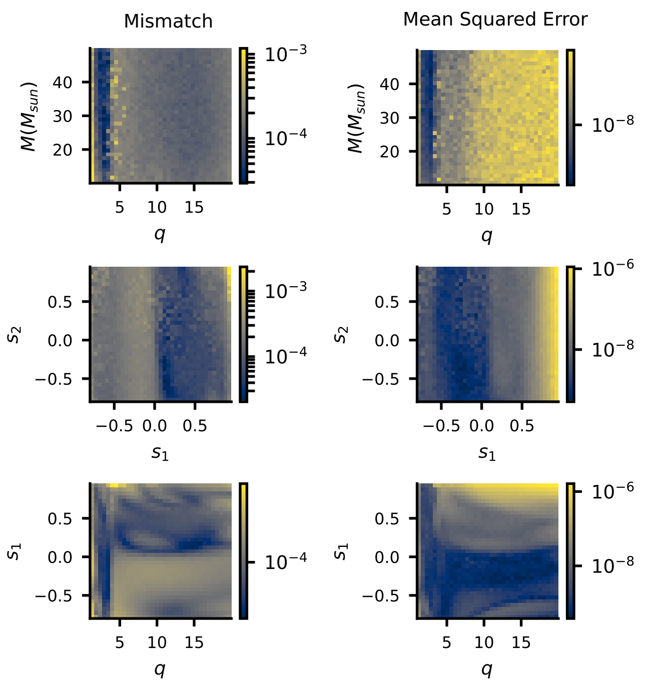

FIG. 6. We report test mismatch and mse for the first PC of the phase, as a function of the orbital parameters $\vartheta = (q, M, s_1, s_2)$. The histograms holds 145061 waveforms, with randomly drawn parameters. In the left column, the color mesh refers to test mismatch; in the right column, we report test mse on the first PC. On the x-y we display values of two physical quantities, each discretized in 35 bins. On top row, we display q vs M dependence. On central row, we present spins dependence s_1 vs s_2. On bottom row, we show q vs s_1 dependence. Each wave starts 8 s before merger. `SS: Do we keep mse column???`

the model. We note also that both $\mathcal{F}$ and mse depend (quite strongly) on q.

In the center line of 6, we displayed the dependence on spins. As long as the s_1 dependence is considered, the most striking feature is the inverse correlation of mismatch and mse for the first phase PC. This means that, being non-leading, spin contribution are not important for the first PC but become dominant at higher order of PCs. Indeed, the values of the first PC are well correlated with mismatch in the case of q. See [?] for a closely related discussion on PCA components and its dependence on physical parameter. `SS: Is it fine to cite this? Is it relevant?`

Apart from this feature, we note that the mismatch tends to grow for low values of s_1: the fit is less reliable in such regions. `SB: [why it is not symmetric in the spins? are they treated differently? Based on the WF morphology/variation I would expect: (i) for up-down spins accuracy similar to nospin, and (ii) for down-down a higher accuracy than up-up ... but maybe this is naive...]`

In the third row of 6, we displayed the dependence on q and s_1, the variables on which the error depends more. We note again the inverse correlation between the mismatch and mse, when considering the s_1 dependence.

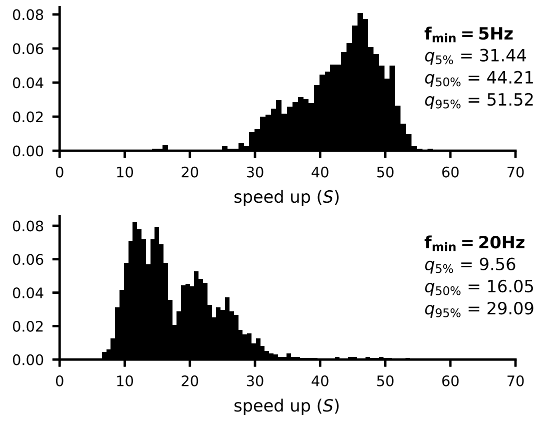

FIG. 7. Histogram for values of the speed up given by `mlgw`, as compared with `TEOBResumS` model, computed on $N = 2000$ test waveforms. Each WF is generated with random physical parameters and has a minimum frequency of 5 Hz (top panel) and 20 Hz (bottom panel). We set a constant total mass $M = 100\,\mathrm{M}_\odot$ and the sampling rate $f_{sam} = 2048$ Hz. We report the median value $q_{50\%}$ as well as the positions $q_{5\%}$ and $q_{95\%}$ of the 5th and 95th percentile. `SS: Are you really sure that two peaks are fine?? Understand this...`

C. Runtime analysis

We now asses the time performances of our model. We are interested to make a comparison with both `TEOBResumS` (the training model) and `SEOBNRv4_ROM` [].

a. Comparison with `TEOBResumS` When dealing with a real detection scenario, we are often interested in generating a WF which starts from a given frequency f_{min}, which is usually set by the detector's sensitivity window. Thus, it is crucial to measure the speed up that our model can provide in performing such task. We define the speed up $\mathcal{S}$ as the ratio between the runtime of benchmark model and the runtime of `mlgw` to produce the a waveform starting from a given f_{min}. Each waveform is produced with constant total mass $M = 100M_\odot$ and random parameters; the WF is sampled at $f_{sam} = 2048$ Hz. We consider the two cases with $f_{min} = 5$ Hz and $f_{min} = 20$ Hz; the first choice refers to the hypotetical lower bound for the sentivity of the Einstein telescope (ET), while the second is close to that of Advanced-LIGO/Virgo. In figure 7 we report the histogram of the measured speed up values for model `TEOBResumS`. `SB: [Have you tried to plot the speedup result as a function of the parameter space, e.g. (q,s1+s2)? e.g. in a 2D scatter plot. Might help indicating which waveform a slower/faster and identifying the original of the two peaks (say, particolarly large q or values of spins)]`

We see that in both cases a substantial speed up is

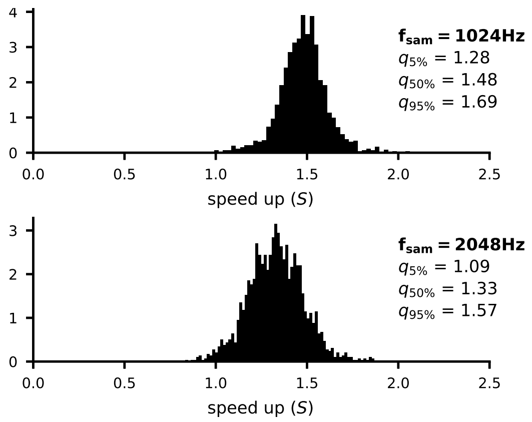

FIG. 8. Histograms for values of the speed up given by `mlgw`, as compared with ROM model `SEOBNRv4_ROM`, computed on $N = 2000$ test waveforms. Each WF is generated with random physical parameters and has a sampling frequency f_{sam} of 1024 Hz (top panel) and 2048 Hz (bottom panel). We set a constant total mass $M = 100\,\mathrm{M}_\odot$ and the same starting frequency $f_{min} = 20\,\mathrm{Hz}$. We report the median value $q_{50\%}$ as well as the positions $q_{5\%}$ and $q_{95\%}$ of the 5th and 95th percentile.

achieved. The speed up is higher for longer WFs [14], making our model particulary convenient for advanced detectors, with a larger sensitivity window.

b. Comparison with SEOBNRv4_ROM In figure 8 we report the histogram of the measured speed up values for model `SEOBNRv4_ROM`, which is now the state of the art for the WF generation time. The comparison is made as above only for $f_{min} = 20\,\mathrm{Hz}$; the ROM has not been built for $f_{min} = 5\,\mathrm{Hz}$. As the ROM model yields WFs in frequency domain, in the time comparison we included a fast Fourier trasform (FFT) of the time domain WF of `mlgw`. This is to ensure that we evaluate the two model at the same conditions. Interestingly, the time taken by the FFT (in the `numpy` implementation) is similar to that required to generate a WF. Thus for a WF in FD, our model cannot be substantially faster, due to the limitation imposed by the FFT.`SB: [How about the interpolation to the uniform grid required before the FFT?]` `SS: I don't get the point here...`

We note the the performances are quite similar to each other. If a lower sampling rate is employed, `mlgw` slighly improves its performances.

It is important to stress that `mlgw` is written in pure Python, while `SEOBNRv4_ROM` is coded in C. In fact, a python code could be easily accelerated (i.e. parellelized, run on GPUs, etc...) with dedicated libraries, thus allowing to push the code performance further. `SS: Is`

[14] This is clearly understood: a longer WF requires more computation for a EOB model, while roughly the same amount of work is done by `mlgw`

Task (for 100 WFs)	CPU time (ms)	
	$N_{grid} = 10^3$	$N_{grid} = 10^5$
Generation of raw WF	6.9 (46.9%)	7 (1.6%)
Interpolation to the user grid	4.5 (30.6%)	194 (45.3%)
Post processing	1.7 (11.6%)	206 (48.1%)
Total	14.7 (100%)	428 (100.0%)

TABLE I. Time taken by different stages of the generation of 100 waveforms; data refers to two different values of N_{grid}. Generation of raw WF refers to the computation of the strain h_{FIT} as produced by a ML model. Interpolation to user grid evaluates the WF on the grid chosen by the user. In the post-processing phase, the dependence on d_L, ι and φ_0 is included.

`it ok now??` Furthermore, `mlgw` is more efficient in reconstructing the waveforms: it requires only $O(5 \cdot 10^3)$ WFs for the training, while in order to build a ROM as many as $O(5 \cdot 10^5)$ WFs are required.

c. Profiling It is interesting to have a knowledge of the time spent by `mlgw` in each stage of the WF generation procedure. We generate 100 waves with random physical parameters and we measure the CPU time spent to execute each basic task. In table I, we compare the results for two values of N_{grid}.

We see that the cost of generating the raw WF does not depend on the number of grid points. On the other hand, the interpolation and the post processing depends on N_{grid} and their cost grows dramatically as the user requires more and more points. It is important to stress that the latter two tasks are slow only because they deal with a large amount of points. Indeed they perform trivial and "quick" operations and their execution relies on well optimized `numpy` routines. If such an amount of datapoints is required, very little space is left for speed up.

V. APPLICATION TO GWTC-1

We test the implementation of our MLGW-TEOBResumS model on the public data from GWTC-1, the first GW catalog []. We trained `mlgw` in the range $\mathcal{P} = [1, 20] \times [-0.8, 0.95] \times [-0.8, 0.95]$ and we set $\tau_{min} = 4\,\mathrm{s}/\mathrm{M}_\odot$. The GWTC-1 catalog consists of 10 BBH systems and a BNS system. Because of the training of the WF model, we do not analyse GW170817, the BNS system, and concentrate exclusively on the BBH signals. Our parameter estimation algorithm is `gwmodel` [] a publicly available infrastructure written in a mixture of `Python` and `cython` that serves as interface for the parallel nested sampling implementation `cpnest` []. The analysis of each BBH system is set up as follows; we choose a total of 2000 Live Points, four parallel MCMC chains with a maximum length of 5000 steps to ensure that each successive sample is independent of the previous. These settings yield an average of ~ 15000 posterior samples and evidence calculations that are accurate to the first decimal digit. For each BBH system we choose prior distributions as described in the GWTC-1 release paper []. Finally, and critically, to ensure that our results can be compared fairly to published ones, we employ the power spectral densities released as part of GWTC-1. No calibration uncertainty model is assumed for these runs.

Figure 9 10 and Table II summarise our results. In Table II we report exclusively summary statistic for the intrinsic parameters of the system. All mass parameters quoted are in the source frame. The redshift of each BBH is estimated from its luminosity distance posterior and converted into a redshift by assuming the cosmological parameters given in Ref. [].

Our results are, in general, extremely similar to what published by the LVK Collaboration. This is reassuring as it validates both the WF model hereby presented as well as the data analysis scheme and sampler implemented[15]. We do note certain interesting aspects that are worth mention; for instance, MLGW tends to recover slightly larger chirp masses and slightly smaller mass ratios compared to GWTC-1. At the same time, the value of χ_{eff} also tends to be slightly larger. These differences are, in general, negligible, but nevertheless indicative of the differences in the physics input in the approximant. The results in Table II have been produced using the TEOBResumS MLGW model. TEOBResumS has a treatment of the angular momenta coupling that is quite different from SEOBNRv4 and hence from the IMRPhenom family of waveforms that, albeit indirectly, is calibrated on SEOBNR. As mentioned, the differences are, at least at the signal-to-noises in GWTC-1, very small, but we find very interesting that a different spin coupling prescription has the potential to impact the inferred spin posteriors, prime example is GW170729, and thus the mass parameters. In light of our findings, we think that the resolution of the spin physics is paramount for an accurate and free from systematics inference of astrophysical properties such as the BBH mass and spin distributions.

VI. FINAL REMARKS AND FUTURE PROSPECTS

SB: [I think this needs a bit more structure. All the material is there it is matter of re-organinzing shortening] e.g. logic following from the paper could be

- Short summary of the key ML ingredients of the method and how to go beyond /improve those

- Short summary of the WF performances and how to improve them, extension of mlgw to modes, precession, and freq-domain, NR data etc

- Short summary of PE application and future prospects

We built a ready-to-use Machine Learning model which generates the (dominant quadrupole) time-domain gravitational wave signal from a binary Black Hole coalescence in the non precessing case. The code is released as the package `mlgw`, publicly available at pypi.org/project/mlgw/. The user might install it with the command `pip install mlgw`. The model consists of a PCA model to reduce the dimensionality of the 22 mode (decomposed in amplitude and phase). A regression (a MoE model) is performed to infer a relation from the orbital paramters to the low dimensional representation of the WF.

It is important to stress that our model is very simple, i.e. it has a very little number of trainable paramters and it is not expensive to train, and flexible, i.e. it works for a large range of paramters and for long waveforms. In [**?**], a ML model for GW generation is built, with similar performances. However, the reduced dimension space is considerably larger ($O(200)$) than ours ($O(10)$). In [**?**], the low dimensional space has a similar dimension $O(30)$ and the ML model achieves a similar performance in the time execution (when computed on a CPU). They manage to achieve a better accuracy $O(2 \cdot 10^{-5})$ but they generate significantly shorter WFs ($f_{min} = 15\,\mathrm{Hz}$ for $M = 60\,\mathrm{M}_\odot$ against $f_{min} \simeq 2.5\,\mathrm{Hz}$) and employ a larger training time ($O(6\,\text{hours})$ against $O(6\,\text{minutes})$).

Remarkably, we discovered that a PCA is able to reproduce a high dimensional wave using a few number of variables. On the other hand, the MoE model is currently it is the “bottleneck” of the model accuracy. For this reason, we explored several alternative regression methods, including neural networks, but none of them showed dramaticaly better performance: perhaps much more computational power and a larger training set are required to improve the fit performances any better.

Despite this, our model shows excellent agreement with the underlying training set. At test time, the median mismatch is $\bar{\mathcal{F}} \sim 5 \cdot 10^{-4}$. Furthermore, a single WF generations takes $0.1 - 5\,\mathrm{ms}$ (depending mostly on the number of grid points required by the user), which is a factor of ~ 40 faster than the underlying training model. `mlgw` slightly outperforms a ROM, which is currently the state-of-the-art for quick generation.

Our ML framework allows for several generalization, which might build a more accurate WF generator. First of all, it is quite straightforward to include higher order modes (HMs) in the WF computation. Different regressions, each for each mode, might be done as we already did for the 22 mode. A future update of `mlgw` is in program. Second, also the precession effects might be included in the model. The precession dynamics could be

[15] However, a full validation of the algorithm is presented in [].

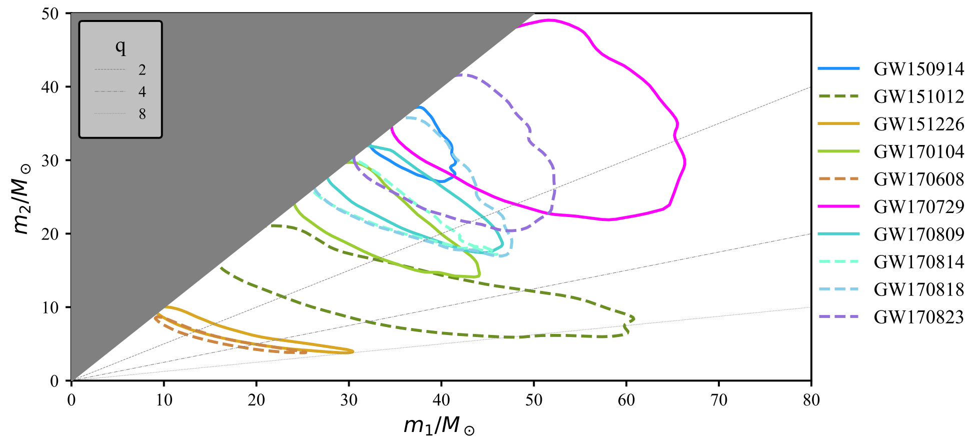

FIG. 9. Two-dimensional posterior distributions for the component masses for all BBH systems in GWTC-1. The contours enclose the 90% credible regions. The figure shall be compared with figure 10 in [?] (left panel).

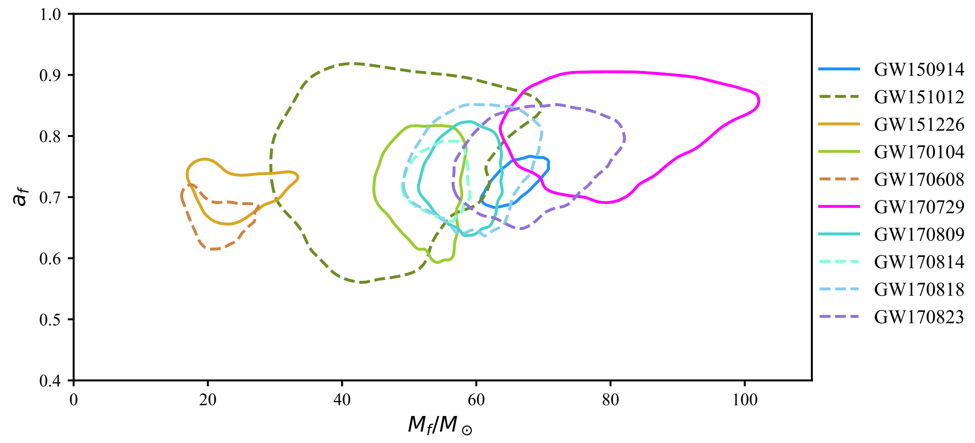

FIG. 10. Two-dimensional posterior distributions for the final masses and spins for all BBH systems in GWTC-1. The contours enclose the 90% credible regions. The figure shall be compared with figure 10 in [?] (right panel).

TABLE II. Summary table for the inferred intrinsic parameters from MLGW and the released GWTC-1 credible intervals. All mass parameters quoted are computed in the source frame, see the text for details of the calculation. For GWTC-1 we report results from Table III, thus coming from averaging over two waveform models. The uncertainties correspond to the 90% credible intervals. SS: What about q in GWTC-1?? Is not in GWTC-1 paper...

	MLGW					GWTC-1				
Event	$m_1/M_\odot$	$m_2/M_\odot$	$\mathcal{M}/M_\odot$	q	χ_{eff}	$m_1/M_\odot$	$m_2/M_\odot$	$\mathcal{M}/M_\odot$	q	χ_{eff}
GW150914	$36.36^{+4.72}_{-2.64}$	$32.64^{+2.93}_{-4.44}$	$29.87^{+1.95}_{-1.50}$	$0.91^{+0.08}_{-0.21}$	$0.14^{+0.10}_{-0.10}$	$35.6^{+4.7}_{-3.1}$	$30.6^{+3.0}_{-4.4}$	$28.6^{+1.7}_{-1.5}$	??	$-0.01^{+0.12}_{-0.13}$
GW151012	$34.51^{+21.37}_{-14.46}$	$11.67^{+6.92}_{-4.46}$	$16.86^{+3.01}_{-2.68}$	$0.33^{+0.56}_{-0.19}$	$0.53^{+0.20}_{-0.33}$	$23.2^{+14.9}_{-5.5}$	$13.6^{+4.1}_{-4.8}$	$15.2^{+2.1}_{-1.2}$	??	$0.05^{+0.32}_{-0.2}$
GW151226	$16.44^{+12.15}_{-5.52}$	$6.38^{+2.86}_{-2.18}$	$8.72^{+0.45}_{-0.27}$	$0.39^{+0.45}_{-0.24}$	$0.32^{+0.24}_{-0.14}$	$13.7^{+8.8}_{-3.2}$	$7.7^{+2.2}_{-2.5}$	$8.9^{+0.3}_{-0.3}$	??	$0.18^{+0.20}_{-0.12}$
GW170104	$31.16^{+10.55}_{-4.77}$	$22.69^{+4.70}_{-6.91}$	$22.83^{+2.64}_{-2.06}$	$0.74^{+0.23}_{-0.35}$	$0.23^{+0.15}_{-0.15}$	$30.8^{+7.3}_{-5.6}$	$20.0^{+4.9}_{-4.6}$	$21.4^{2.2}_{-1.8}$	??	$-0.04^{+0.17}_{-0.21}$
GW170608	$15.45^{+7.60}_{-5.05}$	$5.61^{+2.32}_{-1.50}$	$7.90^{+0.25}_{-0.17}$	$0.36^{+0.40}_{-0.18}$	$0.25^{+0.20}_{-0.17}$	$11.0^{+5.5}_{-1.7}$	$7.6^{+1.4}_{-2.2}$	$7.9^{+0.2}_{-0.2}$	??	$0.03^{+0.19}_{-0.07}$
GW170729	$50.04^{+13.97}_{-9.98}$	$34.78^{+9.78}_{-9.73}$	$35.73^{+7.08}_{-5.08}$	$0.71^{+0.26}_{-0.28}$	$0.51^{+0.18}_{-0.23}$	$50.2^{+16.2}_{-10.2}$	$34.0^{+9.1}_{-10.0}$	$35.4^{6.5}_{-4.8}$	??	$0.37^{+0.21}_{-0.25}$
GW170809	$35.35^{+8.88}_{-5.43}$	$25.17^{+4.81}_{-5.91}$	$25.69^{+2.35}_{-1.74}$	$0.72^{+0.25}_{-0.27}$	$0.24^{+0.16}_{-0.14}$	$35.0^{+8.3}_{-5.9}$	$23.8^{+5.1}_{-5.2}$	$24.9^{+2.1}_{-1.7}$		$0.08^{+0.17}_{-0.17}$
GW170814	$31.35^{+10.70}_{-3.48}$	$25.24^{+3.16}_{-6.40}$	$24.40^{+1.50}_{-1.29}$	$0.81^{+0.17}_{-0.36}$	$0.19^{+0.12}_{-0.11}$	$30.6^{+5.6}_{-3.0}$	$25.2^{+2.8}_{-4.0}$	$24.1^{+1.4}_{-1.1}$	??	$0.06^{+0.12}_{-0.12}$
GW170818	$34.62^{+11.61}_{-5.18}$	$27.09^{+5.59}_{-7.80}$	$26.34^{+4.03}_{-2.77}$	$0.80^{+0.18}_{-0.37}$	$0.28^{+0.20}_{-0.19}$	$35.4^{+7.5}_{-4.7}$	$26.7^{+4.3}_{-5.2}$	$26.5^{+2.1}_{-1.7}$	??	$-0.09^{+0.18}_{-0.21}$
GW170823	$40.69^{+10.21}_{-6.68}$	$31.17^{+7.13}_{-8.09}$	$30.63^{+5.01}_{-3.75}$	$0.78^{+0.19}_{-0.30}$	$0.31^{+0.18}_{-0.20}$	$39.5^{+11.2}_{-6.7}$	$29.0^{+6.7}_{-7.8}$	$29.2^{+4.6}_{-3.6}$	??	$0.09^{+0.22}_{-0.26}$

inserted as a single spin parameter s_P [?] and the WF depencence on s_P can be fitted together with the other orbital parameters.

Furthermore, our model could be trained on the publicly available NR waveforms catalogs (see e.g. [? ? ?] and it would provide the best generalization of the numerical waveform, dispensing with the EOB models altogether. Unfortunately, at the moment there are too few NR waveforms ($O(10^2)$) available to perform a reliable training[16] and the improvement shall wait until enough NR waveforms are available. Moreover, NR waveforms are too short to be used as they are and probably, an extension (e.g. by hybridization with EOB waveforms) towards the early inspiral is needed to compute any kind of NR-based surrogate model.

Lastly, we expect our ML approach to work for every kind of source for which a training set of waveforms is available. Machine learning models to generate WFs might be crucial in the future, where signals from a number of different sources are expected to be detected. In that scenario, a parameter estimation must be able to detect among different source and this will require a lot of computational work. Speed up will be more pressing.

Our work opens up interesting opportunities in GW data analysis (seaches and parameter estimation), both because of its speed and of the closed form expression for the WF.

Due to its speed, `mlgw` could be employed for a systematic comparison between different waveform models, directly on data. By training (and the training procedure is also quick) `mlgw` with different waveform models, it will be possible to compare their predictions on several observed events. This could allow to detect systematic biases or to prefer a model over another by means of Bayesian model selection (i.e. by comparing different model evidences). We started this program by analysing GWTC-1 with `mlgw` trained on `TEOBResumS` and highlited some differences in the predictions as compared to the published results (see section V). Future work might repeat such an analysis on other models or with more observations.

Furthermore, as shown in fig. 7, the model is most useful whenever a long waveform is required: in such case, the speed-up gets even more substantial. This is crucial for the detection of low frequency signals, as is the case for ET. The analysis of such signals can be performed in the same time required to deal with shorter signals: it will become feasible, even with a small amount of resourches, *without any loss of WF quality.*

A closed form expression for the gradients of the waveform with respect to the orbital parameters m_1, m_2, s_1, s_2 [17] could give an advantage on the parameter estimation procedure by using the Hamiltonian MonteCarlo (HM). HM [?] [?] is a variant of Markov chain Montecarlo, which employs gradients of the likelihood (which depends on the gradient of the waveform) to perform an effective sampling of the posterior distibution. The sampling chain conveges faster to the steady state by "find quickly" the high density regions, thus offering a speed up of the PE.

Another option, so far never explored, is to use the gradients of the WF for a fast exploration of the likelihood landscape. With any gradient based optimizer, it should be easy to jump to a *local* maximum of the likelihood. Such information might be helpful to reliably locate a *global* maximum of the likelihood. The ability to quickly locate a global maximum could speed up the searches and the parameter estimation.

`SS: Frase a effetto finale??`

---

[16] At least $O(5 \cdot 10^3)$ waveforms are required.

[17] The computation of gradient is already included in the `mlgw` package.