mlgw NN

Tim Grimbergen,[1] Stefano Schmidt,[1,2,*] Chinmay Kalaghatgi,[1,2] and Chris van den Broeck[1,2]

[1] *Institute for Gravitational and Subatomic Physics (GRASP), Utrecht University, Princetonplein 1, 3584 CC Utrecht, The Netherlands*

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

WRITEME

I. INTRODUCTION

S: TODO:

- Abstract
- Citations

With almost a hundred of confirmed detections, Gravitational Waves (GW) astronomy is entering a mature state, where many loud GW events will force the scientific community to develop a fast analysis to deliver precision measurement. The recent transient catalogue GWTC-3 [] is the latest achievment of the effort carried on by the LIGO-Virgo-KAGRA collaboration and it relies on both instrument and data analysis development.

A crucial element of the data analysis is the ability to generate accurate prediction of the GW signal emitted by a Binary Black Hole (BBH) system. Such waveforms are used for the expensive bayesian estimation of the parameters characterizing a BBH: the analysis of a single event requires the online generation of up to billions of waveforms. As we move towards the next generation of detectors, it is mandatory to deploy accurate waveform models, that are fast and, at the same time, incorporate many physics effects. Failing to do this may lead to systematic errors in the parameter recovery, due to a bad modelling of the source. This is challenging since, speed and accuracy are often at trade.

One important piece for a realistic BBH signal is the inclusion of the Higher Order Modes (HMs) of the multipole expansion of the waveform. For mass symmetric systems, the leading mode is orders of magnitude larger and including HMs doesn't affect the parameter estimation. On the other hand, it has been shown [] that HMs are observable in very asymmetric binary systems, hence the need to include them to avoid any bias.

Throughout decades of developments, two families of models have been developed, both being able to incorporate HMs. One family relies on the Effective One Body (EOB) formalism, which maps the complicated general relativistic binary system into a problem governed by an effective Hamiltonian. EOB models tend to be accurate but are quite costly to evaluate, since for each waveform one needs to solve the hamiltonian equation of motion. On the other hand, the phenomenological waveforms are based on analytical expressions (within the post-newtonian formalism). They tend to be faster to evaluate, but don't achieve the same accuracy. Both families, EOB and phenomenological, need to be calibrated with Numerical Relativity waveforms, computed by solving directly the Einstein equations. The calibration makes sure that a model retains its accuracy even close to merger, where an approximate treatment, such as the post-newtonian or EOB formalism are not applicable anymore.

Besides the standard families, surrogate waveform models have been developed for years with the intent of reproducing the output of a target model and of making feasible the usage of the underlying model. Several surrogates have been developed to accelerate several EOB models [1], even including HMs. While traditional surrogate models [] build an empirical interpolant on the waveform space, a more modern approach relies on performing a regression using Machine Learning techniques [].

Among others, [2] introduced a Machine Learning surrogate model, based on a dimensionality reduction scheme followed by a regression. In this work, we extend this model to HMs and we improve the accuracy of the regression. Our model marks a step towards the development of a fast, yet precise, waveform model by achieving state-of-the-art accuracy and speed, and will enable the accurate analysis of the next generation detectors' data.

We train our model on the widely used approximant `SEOBNRv4PHM` and we achieve a T: This is just the average mismatch then, right? S: Yes, it's just to quote a number that people can remeber easily ??% faithfulness when averaged across a wide range in the parameter space. Our experiments showed that our model offers a substantial speed up with respect to the original model, matching the speed of the state-of-the-art surrogate models.

This paper is organized as follows. In Sec. II we introduce the details of the model presented here, stressing the differences with the model in [2]. Sec. III is devoted to the validation of our model: we will motivate our choice of several hyperparameters and perform an accuracy and speed study. In Sec. IV, we present some final remarks and highlight future perspective.

* s.schmidt@uu.nl

II. BUILDING THE MODEL

A non-precessing BBH can be described by four *intrisic* parameters, which specify the two BH masses m_1 and m_2 and the z-component of the two spin s_1 and s_2. Moreover, since the total mass $M = m_1 + m_2$ sets an overall amplitude scaling, a non-precessing BBH signal only depends on the mass ratio $q = m_1/m_2 \geq 1$ as well as on the spins. We may refer to these parameter as $\boldsymbol{\vartheta} = (q, s_1, s_2)$. Besides the masses and spins, the gravitational wave emitted by the system depends also on luminosity distance to the source d_L, the inclination angle ι of the source and the reference phase φ_0: we call them *extrinsic*.

As it is standard, we expand the angular dependence on ι, φ_0 of the *complex* waveform $h(t)$ in terms of a sum of spin -2 spherical harmonics. A GW is then parameterized[1] as [3]:

$$\begin{aligned} h(t; d_L, \iota, \varphi_0, m_1, m_2, s_1, s_2) &= h_+ + i h_\times \\ &= \frac{G}{c^2}\frac{M}{d_L} \sum_{\ell=2}^{\infty} \sum_{m=-\ell}^{\ell} {}^{-2}Y_{\ell m}(\iota, \varphi_0) h_{\ell m}(t/M; \boldsymbol{\vartheta}) \end{aligned} \quad (1)$$

where we refer to the functions $h_{\ell m}(t; \boldsymbol{\vartheta})$ as *modes* of the waveform. We note that that, for non-precessing systems, $h_{\ell m} = (-1)^\ell h^*_{\ell -m}$, hence we will only consider modes with $m > 0$.

The mode $(\ell, m) = (2, 2)$ is by far the largest in amplitude, hence it is often referred to as the *dominant mode*. The other sub-dominant modes are few orders of magnitude smaller in amplitude and become more relevant (and measurable!) for high mass ratios [].

In this work, we introduce a Machine Learning model to perform a regression

$$(q, s_1, s_2) \longmapsto h_{\ell m}(t; \boldsymbol{\vartheta}) \quad (2)$$

for each mode (ℓ, m). The regression is designed to reproduce waveforms from a given dataset; such waveforms can be generated by *any* time-domain approximant.

We decompose each mode in amplitude and phase

$$h_{\ell m}(t; \boldsymbol{\vartheta}) = A_{\ell m}(t; \boldsymbol{\vartheta}) e^{i\phi_{\ell m}(t; \boldsymbol{\vartheta})} \quad (3)$$

and, for each mode, we perform a regression for amplitude and phase separately. The regression scheme closely follows [2] and relies on:

(a) a suitable vector representation of the regression target by choosing a fixed time grid

(b) a Principal Component Analysis (PCA) model to reduce the dimensionality of each waveform

(c) an Artificial Neural Network (ANN) regression to learn the dependence on $\boldsymbol{\vartheta}$ of the reduced waveform

While the first two elements are unchanged from the previous work, the ANN regression is first introduced here. Indeed a NN has more representation power than the Mixture of Experts (MoE) model [4], used in [2]: the change was needed to achieve better accuracy for the model.

A. Dataset creation

To construct a dataset, we follow [2] and we set a dimensionless time grid. We construct the grid by setting D points equally spaced in τ^α, where τ is the physical time scaled by the total mass of the system M: $\tau = t/M$. Using the findings of [2], we set $D = 2000$ and $\alpha = 0.5$. This is a good compromise between the need of having a faithful representation of the waveform (which requires a large grid) and the need of having a compact model (which points to a sparse grid).

The starting point of the grid τ_0 sets the length of the waveform that our model is able to generate. We choose $\tau_0 = 2\,\mathrm{s/M}_\odot$ and we populate the dataset with 68000 waveforms.

To make sure that the distribution of q is skewed towards towards the boundaries, where the regression is less accurate, we sample the mass ratio q in the range $[1, 10]$ with the following procedure:

- We sample $q_1, \ldots, q_5 \sim \mathcal{U}_{[1,10]}$
- We sample $x \sim \mathcal{U}_{[0,1]}$
- We select q, based on the value of x:
  - If $x \in [0, 0.3)$, $\min q_1, \ldots, q_5$
  - If $x \in [0.3, 0.8)$, q_1
  - If $x \in [0.8, 1]$, $\max q_1, \ldots, q_5$

where $\mathcal{U}_{[a,b]}$ is the uniform distribution in $[a, b]$. The spins are drawn uniformly in the range $[-0.99, 0.99]$. S: This mechanism is complicated but it's late to change it

Once a time grid is set, we evaluate all the modes (amplitude and phase) on the time grid and represent them as vectors in $\mathbb{R}^D$. We then create a dataset $\{X, Y\}$ of N elements. Each row of the dataset is of the form:

$$X = [q, s_1, s_2] \quad (4)$$

$$Y = [\boldsymbol{A}^T_{\ell m}, \boldsymbol{\phi}^T_{\ell m}, \ldots] \quad (5)$$

The dataset Y gathers the amplitude and phase for the different modes in the dataset. We include all the modes available in `SEOBNRv4PHM`: $(\ell, m) = \{(2, 2), (2, 1), (3, 3), (4, 4), (5, 5)\}$.

In what follows we will refer to any of the vectors $\boldsymbol{A}_{\ell m}$ or $\boldsymbol{\phi}_{\ell m}$ as $\boldsymbol{f}$. Note that we use the same grid for all the modes.

[1] Such parameterization is particularly convenient as it separates the waveform dependence over intrinsic and extrinsic parameters.

B. Dimensionality reduction

It is unfeasibile to perform a regression targeting a large dimensional vector such as $\boldsymbol{f} \in \mathbb{R}^D$. For this reason, in [2] we introduced a Principal Component Analysis (PCA) dimensionality reduction scheme. It is an *approximately* invertible linear mapping between a vector $\boldsymbol{f} \in \mathbb{R}^D$ in a large dimensional space to lower dimensional vector $\boldsymbol{g} \in \mathbb{R}^K$:

$$\mathbf{g} = H(\mathbf{f} - \boldsymbol{\mu}) \tag{6}$$

$$\hat{\mathbf{f}} = H^T \mathbf{g} + \boldsymbol{\mu} \tag{7}$$

where $\boldsymbol{\mu} \in \mathbb{R}^D$ and H is a $K \times D$ matrix. The rows $H_{i:}$ of H, also called *Principal Components* (PC), form an orthonormal set of vector, i.e. it holds $\sum_{k=1}^{D} H_{ik} H_{kj} = \delta_{ij}$. The PCs are the first K eigenvectors of the $D \times D$ covariance matrix of the dataset, as described in [5, Sec. 12].

The mapping is only approximately invertible, in the sense that $\hat{\mathbf{f}}$ is only an approximation of the high dimensional vector $\mathbf{f}$. The goodness of the approximation is controlled by the number K of PCs considered: the more PCs, the most accurate is the reconstruction of $\mathbf{f}$. S: I added a paragraph to clarify the introduction of $\hat{\mathbf{f}}$. Does it look good?

One can have a deeper insight on PCA considering the following formula for the reconstructed vector $\mathbf{f}$ (setting $\boldsymbol{\mu} = 0$ without loss of generality):

$$\mathbf{f} = \sum_{i=0}^{K-1} \langle \mathbf{f} | H_{i:} \rangle \; H_{i:} \tag{8}$$

where $\langle \cdot | \cdot \rangle$ is the usual scalar product.

Since less important PCs are more orthogonal to data, the typical magnitude of $g_i = \langle \mathbf{f} | H_{i:} \rangle$ decreases as i increases.[2]. As a consequence, the regression for a lower order PC needs to be more accurate than the one for the higher order PC. This will be taken care by a suitable choice for the loss function for the regression (see next section).

Following [2], in this work we employ 6 PCA components for the phase model and 4 for the amplitude.

C. Neural network regression

T: General remark: We should choose a convention whether we index the first principal component with a 0 or a 1. Right now we are using them both (for phase)... S: You're right. I would start numbering from index zero, but would call it *first* PC. Do you think it's confusing? An Artificial Neural Network (ANN) is a popular regression model, consisting of a powerful parametric function, whose parameters (or weights), when properly set, can represent a large variety of relations between input and output. An ANN is built by stacking together N_{L} layers in such a way that the output of a layer is the input of the following layer. Each layer is a function $L : \mathbb{R}^{D'} \to \mathbb{R}^{D''}$ and has the following functional form

$$\boldsymbol{y} = a(W'_{''} \boldsymbol{x}) \tag{9}$$

Where $W'_{''}$ is a $D'' \times D'$ matrix and $a : \mathbb{R} \to \mathbb{R}$ is an activation function that acts elementwise on the vector $W'_{''}\boldsymbol{x}$. Each component y_i of the output of the layer is called a node and the number of nodes is a tunable parameter, controlling the representative power of the layer.

An ANN $\mathcal{N}$ is obtained by composing N_{L} different layers (each with a suitable number of nodes):

$$\mathcal{N}_W = L_{N_{\mathrm{L}}} \circ \ldots \circ L_2 \circ L_1 \tag{10}$$

where we denote by W the set of all the parameters the ANN depends on.

The number of layers, together with the number of nodes per layer are hyperparameters that need to be carefully chosen, to balance model accuracy and model complexity. Another important choice is the activation function: several possible choices are possible, the most popular being the *sigmoid*, the hyperbolic tangent or the so called ReLU function. In our work, we consider the sigmoid function between all layers, except for the very last layer which has linear/identity activation so that negative values are also possible.

Once the ANN is set up, we need to set its weights to the values that achieve our regression task. This procedure is called training, where we minimize a loss function with respect to the weights $\boldsymbol{W}$ of the model. The loss function depends on the dataset at hand $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_i$. Mathematically, the weights are given by:

$$\boldsymbol{W} = \arg\min_{W} \mathcal{L}(W; \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_i) \tag{11}$$

The minimization of the loss function is performed by stochastic gradient descent (SGD), as implemented by the `Nadam` algorithm [6, 7]. The optimization relies on the gradients $\partial_W \mathcal{L}$ of the loss function, computed through the backpropagation algorithm [].

To perform our regression $\theta \longmapsto \boldsymbol{g}$, we employ an ensemble of networks that suitably combined delivers accurate results. To improve the representative power or the ANN, we employ feature augmentation on the vector $\boldsymbol{\vartheta} = (q, s_1, s_2)$, effectively using the augmented vector $\tilde{\boldsymbol{\vartheta}}$ as input for the regression. Although different ANN's will need different features, we will for convenience abuse

[2] For this reason PCA can be seen as a perturbative expansion on the basis vectors $H_{i:}$, where the accuracy is roughly measured by the eigenvalues of the first neglected PC. Increasing the number K of PCs considered increases the accuracy of the model (but also the complexity of the model).

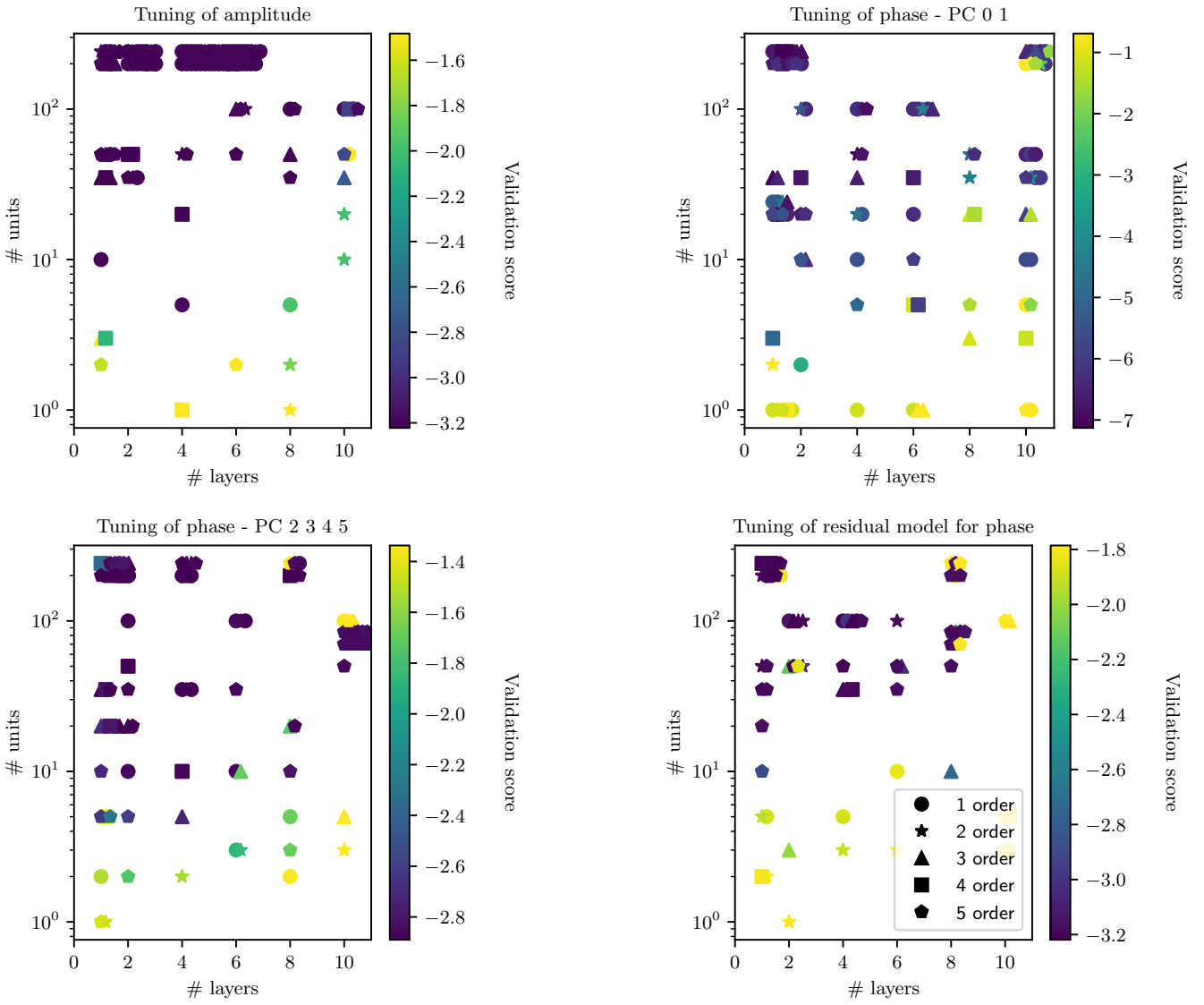

FIG. 1. Results from the validation of our ANN models, using the $\ell, m = 2, 2$ mode dataset. We tune the number of layers and the number of features per layer, together with the features and the polynomial order for the data augmentation. Each panel in the figure refers to a different ANN, taking care of different parts of the regression, as described in Sec. II C. For each regression, we train 100 ANNs with different choices of hyperparameters. Each point in the plot, refers to a trained network and it is colored with the logarithm of the validation score. Note that we do not report the features used for data augmentation, so that the plot is degenarate in this quantity.

the notation $\tilde{\boldsymbol{\vartheta}}$ to any augmented vector. Indeed, the features to add need to be chosen with a validation process: this will be discussed in the next section.

Before the training, the regression targets $\boldsymbol{y}_i$ are scaled such that $\boldsymbol{y}_i \to \frac{\boldsymbol{y}_i}{\boldsymbol{w}}$, where $\boldsymbol{w}$ keeps the maximum of $|\boldsymbol{y}_i|$ along each axis. In this way all the regression targets span the same order or magnitude, making the regression task easier.

For the amplitude $\boldsymbol{A}_{\ell m}$ of each mode, we employ a single ANN $\mathcal{N}_{A_{\ell m}}$ that predicts the first four PCA components. The predicted amplitude $\hat{\boldsymbol{A}}_{\ell m}$, including the PCA reconstruction, has the following form:

$$\hat{\boldsymbol{A}}_{\ell m}(\boldsymbol{\vartheta}) = \boldsymbol{\mu}_{A_{\ell m}} + H^T_{A_{\ell m}} \mathcal{N}_{A_{\ell m}}(\tilde{\boldsymbol{\vartheta}}) \tag{12}$$

For the phase $\boldsymbol{\phi}_{\ell m}$, we employ one ANN $\mathcal{N}_{\phi_{\ell m}\text{- }01}$ to predict only the first two PCA components. Another ANN will take care of the remaining components $\mathcal{N}_{\phi_{\ell m}\text{- }2345}$. S: Added in the PCA section a remark on how many PCs we are using, citing the previous work On top of this, we build an additional ANN $\mathcal{N}_{\phi_{\ell m}\text{- residual}}$ to target the residual of the predictions of $\mathcal{N}_{\phi_{\ell m}\text{- }01}$ T: Is the term "residual" a well-known term (or clear enough from context)

or does it require more explanation? S: I would say it's pretty clear, but I'm happy to add a clarification, if needed. . The scheme make sure that the first two PCs are predicted with much larger accuracy than the others. Indeed, the reconstructed WF depends largely on the first two components and a small fractional error can potentially have a large impact on the overall accuracy.

The predicted phase $\hat{\boldsymbol{\phi}}_{\ell m}$ is then given by:

$$\hat{\boldsymbol{\phi}}_{\ell m}(\boldsymbol{\vartheta}) = \boldsymbol{\mu}_{\phi_{\ell m}} + H^T_{\phi_{\ell m}} \begin{pmatrix} \mathcal{N}_{\phi_{\ell m}\text{- }01}(\tilde{\boldsymbol{\vartheta}}) + \mathcal{N}_{\phi_{\ell m}\text{- residual}}(\tilde{\boldsymbol{\vartheta}}) \\ \mathcal{N}_{\phi_{\ell m}\text{- }2345}(\tilde{\boldsymbol{\vartheta}}) \end{pmatrix}. \tag{13}$$

We train our model using the PCA dataset, obtained by PCA reducing the training set. Each ANN is trained using the following loss function: T: I know in the definition of the custom loss function the w for the weights is inside the square, but (at least in the fitNN function that I wrote) I think we take the square root of $\boldsymbol{w}$ as input of the loss weights, so that the $\boldsymbol{w}$ in the definition below should be outside of the square... S: You're right. But in this case, the heuristics of why we chose such weights is less clear. Do you have a good explaination of why this works?

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\Big(\big(\mathcal{N}(\boldsymbol{\vartheta}_i) - \boldsymbol{y}_i\big)\Big)^2 \boldsymbol{w} \tag{14}$$

where $\boldsymbol{y}_i$ is the (scaled) regression target of each network and $\boldsymbol{w} \in \mathbb{R}^K$ takes into account the fact that different PCs have different orders of magnitude.

The network is implemented and trained using the python package `keras` [8], built on `tensorflow` backend [9].

III. PERFORMANCE STUDY

In this section, we first study how the model performance depends on the different choices of hyperparameters (network architecture, learning rate, features etc...). The architecture details of the model (chosen after hyperparameters tuning) are reported in Tab. III A. We then evaluate the faithfulness of our model and report the speed up that we obtain when using `mlgw-NN` instead of the training model `SEOBNRv4PHM`.

To measure the discrepancy between two waveforms h_1, h_2, we define a scalar product:

$$(h_1|h_2) = 4\Re\int_{-\infty}^{\infty} \mathrm{d}f\, \frac{\tilde{h}_1^*(f)\tilde{h}_2(f)}{S_n(f)} \tag{15}$$

where $\tilde{}$ denotes the Fourier transform and $S_n(f)$ is the Power Spectral Density (PSD) of the detector's noise. We can use the scalar product to normalize a waveform: we denote by $\hat{h} = \frac{h}{\sqrt{(h|h)}}$ the normalized waveform h.

To measure the discrepancy between two individual modes $h^1_{\ell m}$ and $h^2_{\ell m}$, we define the *match* $\mathcal{M}$:

$$\mathcal{M} = \max_t (\hat{h}^1_{\ell m}|\hat{h}^2_{\ell m} e^{i2\pi f t}) \tag{16}$$

where $he^{i2\pi ft}$ denotes (with a slight abuse of notation) h translated in time by a factor of t. We call *mismatch* the quantity $\mathcal{F} = 1 - \mathcal{M}$.

The match defined above amounts to the the search statistics being used for matched filtering searches of non-precessing/non-HM signals []. A different stastics is needed to search for HM signals, hence the match defined above is not suitable to compare two different waveforms with HM content Eq. (1). In this case, we need to compare the two polarizations h_+, $h_\times$ of a waveform with a signal s observed at the detector:

$$s = F_+ h_+ + F_\times h_\times \tag{17}$$

where $F_+, F_\times$ are called antenna pattern functions, depending on the sky location of the source and on the polarization angle [].

We are then ready to introduce the *symphony match*[3] between a signal s and a waveform h:

$$\mathcal{M}_{\text{sym}} = \max_t \frac{(\hat{s}|\hat{h}_+)^2 + (\hat{s}|\hat{h}_\times)^2 - 2(\hat{h}_\times|\hat{h}_+)(\hat{s}|\hat{h}_+)(\hat{s}|\hat{h}_\times)}{1 - (\hat{h}_\times|\hat{h}_+)^2} \tag{18}$$

note that the $\mathcal{M}_{\text{sym}}$ depends on the signal s, hence it depends on the sky location and polarization angle. As above, we define the symphony mismatch as $\mathcal{F}_{\text{sym}} = 1 - \mathcal{M}_{\text{sym}}$. S: I didn't want to go into too many details here but maybe the whole discussion is unclear and needs more details

In what follows, we always use a constant (i.e. flat) PSD. While this certainly does not correspond to any actual detector, it makes sure that all the frequencies are weighted equally, hence giving a detector agnostic measure of the mismatch. S: Does it make sense? Or rather should we use a PSD?

A. Hyperparameters tuning

T: I believe you used the exact same plot in Fig. 3 for (q, s_{1z}) and (q, s_{2z}), or is it a super coincidence? :) S: You are so right! There was a bug in the code that produced the json of the results. Will need to run again :D

The performance of the model depends on a number of crucial choices about some non-trainable parameters, usually called hyperparameters. The hyperparameters usually defines the architecture of the ANN as well as

[3] The name comes from the paper [] first introducing it

Network	`n-layers`	`units`	`features`	`order`
$\mathcal{N}_{A_{\ell m}}$	1	35	$\mathcal{M}_c, \chi_{\rm eff}$	1
$\mathcal{N}_{\phi_{\ell m}\text{- 01}}$	2	50	$\mathcal{M}_c, \eta, \log q, \chi_{\rm eff}$	3
$\mathcal{N}_{\phi_{\ell m}\text{- 2345}}$	1	50	$\mathcal{M}_c, \eta, \log q, \chi_{\rm eff}$	1
$\mathcal{N}_{\phi_{\ell m}\text{- residual}}$	5	50	$\mathcal{M}_c, \eta, \log q, \chi_{\rm eff}$	2

TABLE I. Architecture of the 4 ANNs employed to generate each mode. For each ANN we report the number of layers and the number of units per layer. We perform data augmentation by adding all the polynomials terms in the chosen features. The architecture have been chosen after hyperparameter tuning (see Fig. 1). Among other features, we use the chirp mass $\mathcal{M}_c = \frac{(m_1 m_2)^{3/5}}{(m_1+m_2)^{1/5}}$, the symmetric mass ratio $\eta = \frac{m_1 m_2}{(m_1+m_2)^2}$ and the effective spin parameter $\chi_{\rm eff} = \frac{m_1 s_{1z} + m_2 s_{2z}}{m_1+m_2}$

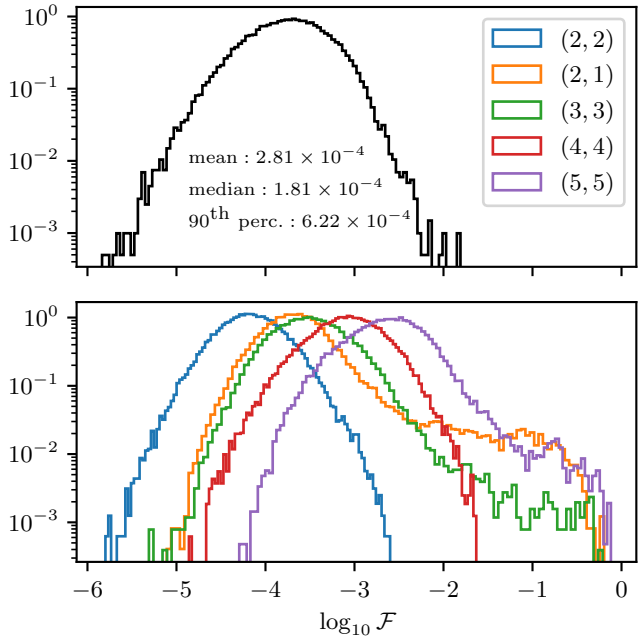

FIG. 2. We report the results of the mismatch between the 50000 test waveforms produced by `mlgw` and by the training model `SEOBNRv4PHM`. In the top panel, we report the histogram of the "symphony" mismatch $F_{\rm sym}$ for the overall waveforms, where we compare the h_+ and $h_\times$ polarizations (see Eq. (8)). For the computation, we set random sky location. We also report the median and the mean mismatch, together with the value of the $90^{\rm th}$ percentile. In the bottom panel, we report the histograms for the mismatches computed mode by mode. The composition of the test set is described in the text.

some parameters relevant to the training. Setting the right values for the hyperparameters is crucial for the ANN performance, as one needs to balance between accuracy and speed: this procedure is called *hyperparameters tuning* and can be done automatically to optimize manual work and to make sure to find a good minimum.

We optimize the following hyperparameters:

- `n-layers`: number of hidden layers in the ANN
- `units`: number of nodes per hidden layer
- `features`: features to use for data augmentation
- `order`: the data will be augmented with all the monomials of the chosen features up the given order

For each of the 4 ANN useful to produce a single mode (see Eq. (12-13)), we train a network for different combinations of hyperparameters. The figure of merit of each hyperparameter choice is the logarithm of the loss function Eq. (14) evaluated on the validation set. For our experiments we only use the dataset of the $(2, 2)$ mode and we employ the package `keras-tuner` [10].

We report our results in Fig. 1, where each combination of hyperparameters tested is represented in the `n-layers-units` plane and colored by the validation scored. We can see that all the four ANN share the same trend: the most effective way to improve regression accuracy is to increase the number of units as opposed to the number of layers. The number of layers is far more important the extra features added and the polynomial order for data augmentation.

Furthermore, we note that the regressions for the amplitude and for the high phase PCs (i.e. components 2,3,4,5) can be performed with a smaller model, compared to the models for the first two PCs of the phase. This can be explained by the fact that most of the physical information is stored in the first two components of the phase, making this an harder regression problem.

In table Tab. III A we report the final hyperparameter choice we made for each of the networks. The architectures are the same across the different modes considered.

As discussed above, we note that models $\mathcal{N}_{A_{\ell m}}$ and $\mathcal{N}_{\phi_{\ell m}\text{- 2345}}$ are very simple, having only one layer and a small polynomial order, while the other ANNs have a more complicated architecture. We note here that an accurate ANN for the residuals of the phase is crucial to obtain a good accuracy: indeed $\mathcal{N}_{\phi_{\ell m}\text{- residual}}$ is the most complex model we employ, meaning that the residual phase dataset is the "hardest" to learn.

B. Accuracy study

To test the accuracy of our model, we generate a test set with 50000 randomly chosen waveforms generated with the training model `SEOBNRv4PHM`. The waveforms masses are characterized by a total mass in the range $[30, 80] M_\odot$, by a mass ratio $q \in [1, 10]$. The spins are chosen in the range $[-0.99, 0.99]$ and the inclination angle ι and reference phase φ_0 are drawn uniformly from a sphere. We sample the starting frequency uniformly in the range $[10, 20] Hz$.

In Fig. 2, we report the histogram of the distribution of the mismatches between `mlgw` and the test waveforms. The upper part refers to the mismatches Eq. (18) computed on the overall waveforms (with sky location sampled uniformly over the sky); the lower box refers

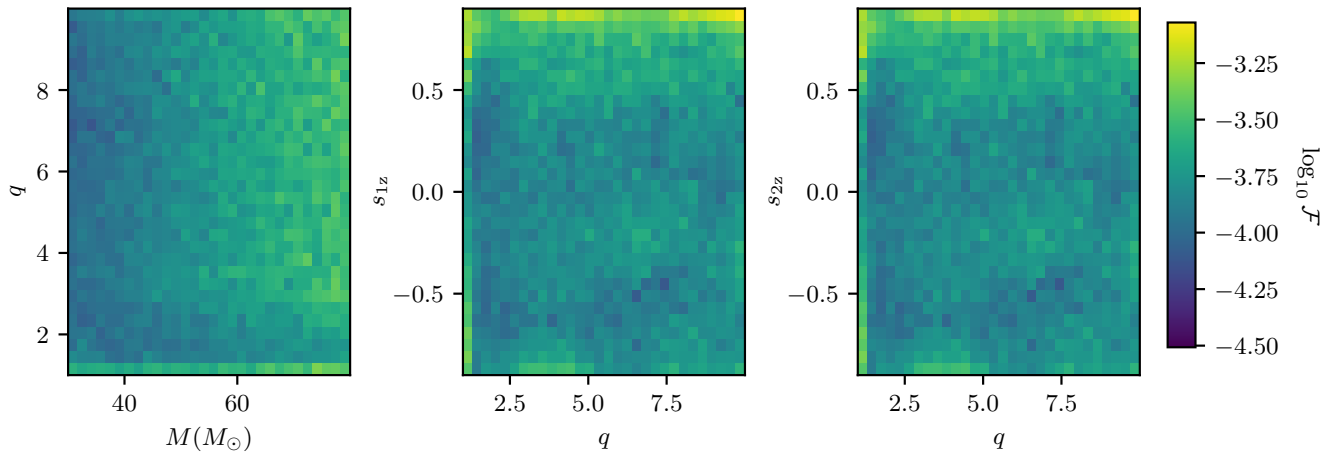

FIG. 3. Depency of the "symphony" mismatch $F_{\rm sym}$ between `mlgw` and the training model `SEOBNRv4PHM`, as a function of some chosen orbital paramers. The mismatch is computed on the 50000 waveforms on the test set described in the text. On the left plot, we display the quantities $M - q$ on the two axis, while on the center and left plot we show on the axis the variables $q - s_{1z}$ and $q - s_{2z}$ respectively. Each bin is colored according to the *average* mismatch and the three plots shares the same color scale. We note that `mlgw`'s faithfulness tends to decreases for low values of q and large positive values of s_{1z} S: This plot will need to be updated - working on it :D

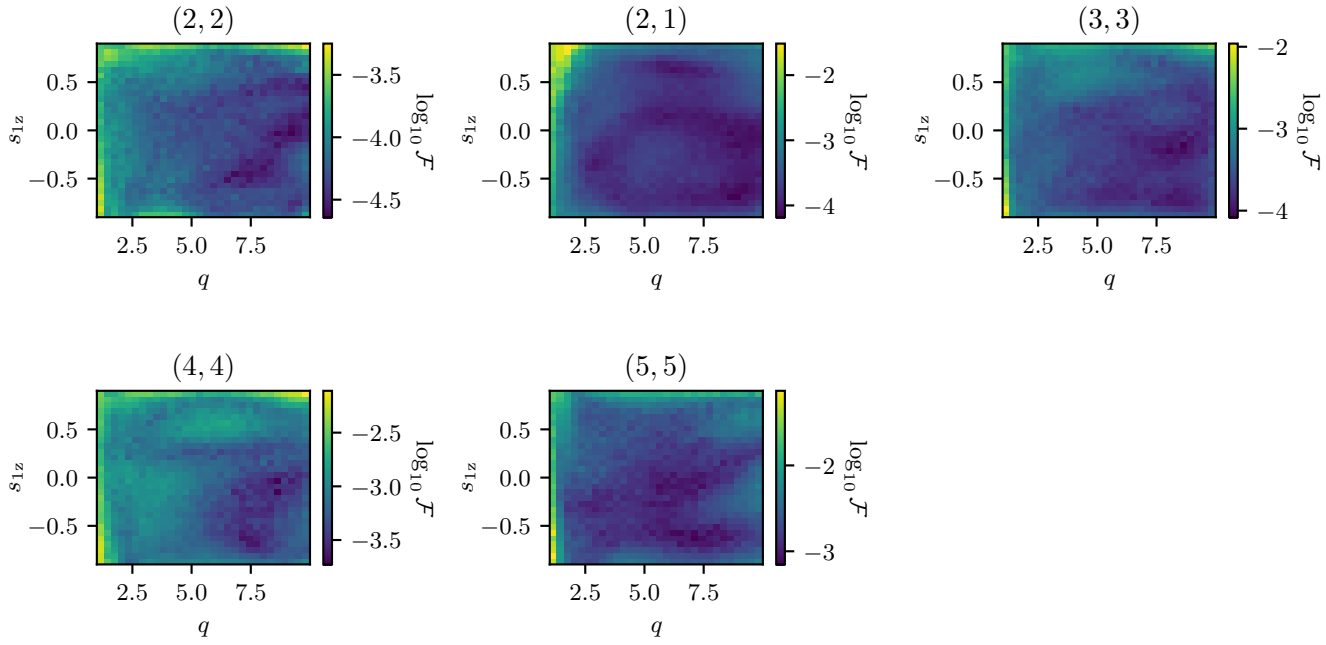

FIG. 4. For each mode, we report the mismatch between `mlgw` and the training model `SEOBNRv4PHM`, as a function of q and s_{1z}. The mismatch is computed on the 50000 waveforms on the test set described in the text. Each bin is colored according to the *average* mismatch. We note that the performance between different modes can vary significantly and in general they decrease for low values of q and high values of spins.

to mismatches computed mode by mode with Eq. (16). T: I think it is a bit weird that the legend of the bottom figure is in the figure above... though this is an efficient way of using the space I guess. S: Yes! The figure above was empty otherwise... Unless you strongly object, I'd keep it as it is.

First of all, we note that the model shows very high faithfulness. With a median value of 2×10^{-4} and with virtually no signals with a "symphony" mismatch higher than 10^{-2}, `mlgw-NN` is indistinguishable from `SEOBNRv4PHM` for the current generation detectors [**?**].

The faithfulness for the $(2,2)$ mode is even higher with no signals with mismatch higher than 2×10^{-3}. On the other hand, the higher order modes are less accurately reproduced than the dominant mode. In particular, for the modes $(2,1),(3,3),(5,5)$ a limited number of waveforms show very high mismatches $\mathcal{O}(1)$.

The cause of the decreased faithfulness for subdominant modes needs more investigation. However, it is likely due to the fact that the physics of subdominant modes is less understood, hence the uncertainties in waveform modeling results in a more noisy regression for the ANN to learn. This problem can be probably mitigated by using a larger network for such modes. Indeed, we tuned the hyperparameters on the $(2,2)$ mode (an "easy" regression target). Performing a tuning on the dataset of higher order modes might reveal that our chosen architecture is not optimal.

In Fig. 3 we report the dependence of the "symphony" mismatch as a function of the different orbital parameters. From the figure, it is manifest that the model has very stable performance across the parameter space. The faithfulness decreases for high positive values of the spins and for mass ratio $q \sim 1$: such cases are at the boundaries of the waveform dataset, hence they correspond to extreme values in the PCA reduction, making harder for the network to predict such cases. Despite this, in such "extreme" regions, the average mismatch is still of the order of 10^{-4}. We also note that for $q \sim 1$, the sub-dominant modes have a vanishing amplitude: a large mismatch in the sub-dominant mode for $q \sim 1$ has very little impact on the overall waveform Eq (1).

In Fig.4, for each mode we report the mismatch as a function of the mass ratio and of s_{1z}. One more time, we can see that the model faithfulness decreases for low mass ratios and for high spins. Moreover, the subdominant modes shows a poorer performance as compared to the dominant one.

C. Timing study

A speed up in the waveform generation is the main motivation to build a ML waveform generator; for this reason it is crucial to assess the gain in waveform generation time. For this reason, we use our test set to measure the ratio between the time to generate a waveform with `SEOBNRv4PHM` and `mlgw`. Our model offers further speed by generating waveform in batches: in this case, some operations are efficiently parallelized and happen more efficiently. We report our findings in Fig. 5.

We achieve a speed up between a factor of 100 and 200, depending on the waveform. T: The sentence ``depending on the length of the waveform" makes sense, but does it need any ``proof"? S: You're right. If we don't provide any dependence of the speed up on the WF length, we cannot make such claim. Since I don't think we want to add another plot (do we?), I changed the sentence into a less controversial statement. For waveform batch generation, the speed up can be twice as much, ranging between 200 and 400. The speed up achieved by `mlgw` is slightly larger than the one obtained by a surrogate model `SEOBNRv4PHMSur` obtained with standard techniques [1].

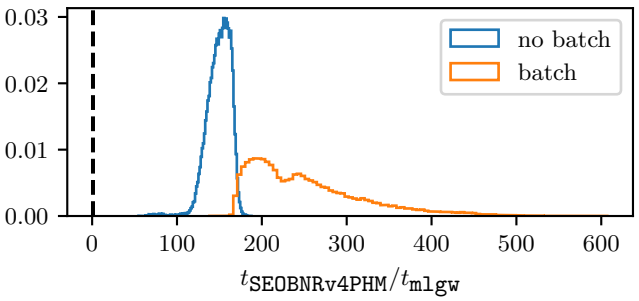

FIG. 5. Speed up provided by `mlgw` over the training model `SEOBNRv4PHM`. In the histogram, we report the ration between the time $t_{\texttt{SEOBNRv4PHM}}$ and the time $t_{\texttt{mlgw}}$ taken by the two models to generate each of the waveform in the test set. We note that `mlgw` offers a speed up between a 100 and 200 with respect to the training model. `mlgw` offers the option to generate waveforms in batches, effectively parallelizing some linear algebra operations. As shown in the plot, the batch generation provides a speed up of roughly a factor of two as compared with the non-parallelized version.

IV. FINAL REMARKS AND FUTURE PROSPECTS

Building on our previous work [2], we generate a ML surrogate model `mlgw-NN` able to reproduce with very high fidelity the output of the widely used approximant `SEOBNRv4PHM`. `mlgw-NN` can generate waveforms in a cuboid $q \times s_{1z} \times s_{2z} = [1,10] \times [-0.9,0.9] \times [-0.9,0.9]$ on a (reduced) time grid of maximum length of $2\,\mathrm{s}/\mathrm{M}_\odot$. Our model offers a two orders of magnitude speed up over the training model, without trading for accuracy, hence it is an attractive alternative for any data analysis application. Our method is fully general and is applicable to any chirp-like gravitational wave signal. To encourage new applications, we release our code (and our trained model) publicly as a python package through the PyPI repository.

Future work should also include precession. This can be achieved by means of the *spin twist* procedure []. It consist on a time dependent rotation of the plane of emission, resulting in a phase and amplitude modulation which approximates the effect of precession. Training an ANN to predict the time dependent rotation is a promising step towards a complete ML surrogate model. S: GENERAL COMMENT: we should stress somewhere, maybe in the intro, that next generation ifos will need surrogate models and

that we don't quite match yet the accuracy. This is the whole motivation of exploring new stuff like we are doing.

While the model is already applicable for most of the parameter estimation problems with current detectors, it is desirable to increase its range of validatity of our model, both in parameter space and in time span.

In principle, such extension should be straightforward using the current architeture. On the other hand, due to an increased complexity of the regression task, probably more flexible architetures should be explored, using layers of different size. This would require a more careful (and computationally expensive!) hyperparameters tuning. In this context, tuning the network separately for each mode can also be beneficial. As noted in Sec. III B, some HMs have worse performance than others, implying that the same architecture may not be valid for the prediction of different modes. Tayloring the architecture to each regression problem could be an elegant way to improve substantially the performance of our model.

These improvements will become mandatory for the next generation detectors [], when fast and reliable waveform models will be needed to mitigate the huge computational cost posed by very long observed waveforms. Our framework is ideal to achieve such ambitious goal. S: I don't feel like adding more will be beneficial. On the other hand, this section is very short T: Maybe we could add a paragraph on different neural network topologies (not the same number of units for each hidden layer) and repeat what we said about improving HMs by using separate tuning? S: Added this: how does this look?

ACKNOWLEDGMENTS

This research has made use of data, software and/or web tools obtained from the Gravitational Wave Open Science Center (https://www.gw-openscience.org), a service of LIGO Laboratory, the LIGO Scientific Collaboration and the Virgo Collaboration. LIGO is funded by the U.S. National Science Foundation. Virgo is funded by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale della Fisica Nucleare (INFN) and the Dutch Nikhef, with contributions by Polish and Hungarian institutes.

[1] B. Gadre, M. Pürrer, S. E. Field, S. Ossokine, and V. Varma, "A fully precessing higher-mode surrogate model of effective-one-body waveforms," 3 2022.

[2] S. Schmidt, M. Breschi, R. Gamba, G. Pagano, P. Rettegno, G. Riemenschneider, S. Bernuzzi, A. Nagar, and W. Del Pozzo, "Machine Learning Gravitational Waves from Binary Black Hole Mergers," *Phys. Rev. D*, vol. 103, no. 4, p. 043020, 2021.

[3] H. Estellés, M. Colleoni, C. García-Quirós, S. Husa, D. Keitel, M. Mateu-Lucena, M. d. L. Planas, and A. Ramos-Buades, "New twists in compact binary waveform modeling: A fast time-domain model for precession," *Phys. Rev. D*, vol. 105, no. 8, p. 084040, 2022.

[4] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.

[5] K. Murphy, *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series, MIT Press, 2012.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[7] T. Dozat, "Incorporating Nesterov Momentum into Adam," in *Proceedings of the 4th International Conference on Learning Representations*, pp. 1–4.

[8] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[10] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, *et al.*, "Kerastuner." `https://github.com/keras-team/keras-tuner`, 2019.