








Cool work in Regression

Simone Albanesi ^{1,*} Marina Berbel ^{2,†} Marco Cavaglia ^{3,‡} Lorena Magaña Zertuche ^{4,§} Miquel Miravet-Tenés ^{5,¶} Dimitra Tseneklidou ^{6,**} and Yanyan Zheng ^{3,††}

¹*Dipartimento di Fisica, Università di Torino & INFN Sezione di Torino, via P. Giuria 1, 10125 Torino, Italy*

²*Departament de Matemàtiques, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain*

³*Institute of Multi-messenger Astrophysics and Cosmology & Physics Department,
Missouri University of Science and Technology, Rolla, MO 65409, USA*

⁴*Department of Physics and Astronomy, University of Mississippi, University, Mississippi 38677, USA*

⁵*Departament d'Astronomia i Astrofísica, Universitat de València, Dr. Moliner 50, 46100, Burjassot (València), Spain*

⁶*Theoretical Astrophysics Department, Eberhard-Karls University of Tübingen, Tübingen 72076, Germany*

(Dated: January 31, 2024)

Abstract goes here.

I. INTRODUCTION

More than seven years ago, the first gravitational wave (GW) from a compact binary coalescence (CBC) was detected by the Laser Interferometer Gravitational-wave Observatory (LIGO) [1]. Ever since that first detection, the LIGO-Virgo-KAGRA (LVK) detector network has brought us to a total of more than 90 confident GW detections [2]. The sources of these detections originate from binary black hole (BBH) systems, binary neutron star (BNS) systems, and neutron star-black hole (NSBH) binaries. With increased detector sensitivity in the upcoming fourth observing run (O4), the LVK is expected to detect as many as one GW signal per day [3]. This will require a quick turn-around time for analyses and results, prompting an increase in automation of searches, data quality algorithms, and parameter estimation (PE) pipelines.

In order to rapidly estimate the intrinsic and extrinsic parameters of CBC signals, such as the masses, spins, and inclination angles, the LVK low-latency detection pipelines use a technique called matched-filtering. This method matches the observed GW signal with a waveform derived from general relativity. The intrinsic parameters of this best-fit waveform are then given as possible true parameters of the binary system detected. However, these parameters may not be the actual parameters of the astrophysical system due in part to waveform systematics but mainly due to detector noise. Moreover, a single event, or injection in our case, may ring multiple triggers corresponding to different templates. This may add a layer of complication. Therefore, for a detection a slower but more accurate PE follow-up is needed. These PE algorithms are robust but can take hours or days to run. This can be problematic when the signal observed comes

from a system that radiates energy in the electromagnetic (EM) spectrum. Recent improvements in Bayesian techniques for PE have reduced the time needed to estimate the source parameters of a GW detection [4, 5] and proposed machine learning (ML) algorithms promise to further reduce this latency by several orders of magnitude [6]. Current online results using Bilby [7] and RapidPE-RIFT [8] have decreased the time required for PE in low latency to be ~ 10 minutes. This result is obtained by suitably narrowing the PE priors and choosing aggressive sample settings. Nevertheless, there is still much to gain by reducing the latency of online PE further. During O4, the LVK is planning to issue preliminary alerts for candidate detections seconds after the merger, and in some cases, an early warning alert with information on sky localization may be issued before [9]. Providing a higher level of accuracy at low latency will allow us to better estimate the time and peak luminosity of the afterglow emission in the case of a coincident EM signal. It will give the LVK astronomy partners an observational head-start.

Although ML regression analysis has been recently applied to detector characterization and data analysis [10–14], in this paper we present its first application in low latency PE. Specifically, we demonstrate that by using Gaussian process regression (GPR) and neural networks (NNs) we can significantly improve pipeline estimates of the masses and spin magnitudes of CBC detections. Once trained, results can be obtained on timescales of ms per event, making regression a powerful tool in low-latency. This allows us to bridge the gap between the speed and accuracy trade-off.

II. BINARY SYSTEM DATA

The dataset represents over 203,000 binary systems comprised of BBHs, BNSs, and NSBHs. The injections of these systems come from the Mock Data Challenge (MDC) conducted during LIGO's second observing run (O2) and are recovered using *GstLAL*, a low-latency detection pipeline. We divide this dataset into training and testing sets consisting of $\sim 142,000$ and $\sim 61,000$ sys-

* simone.albanesi@edu.unito.it

† mberbel@mat.uab.cat

‡ cavagliam@mst.edu

§ lmaganaz@go.olemiss.edu

¶ miquel.miravet@uv.es

** dimitra.tseneklidou@uni-tuebingen.de

†† zytfc@umsystem.edu

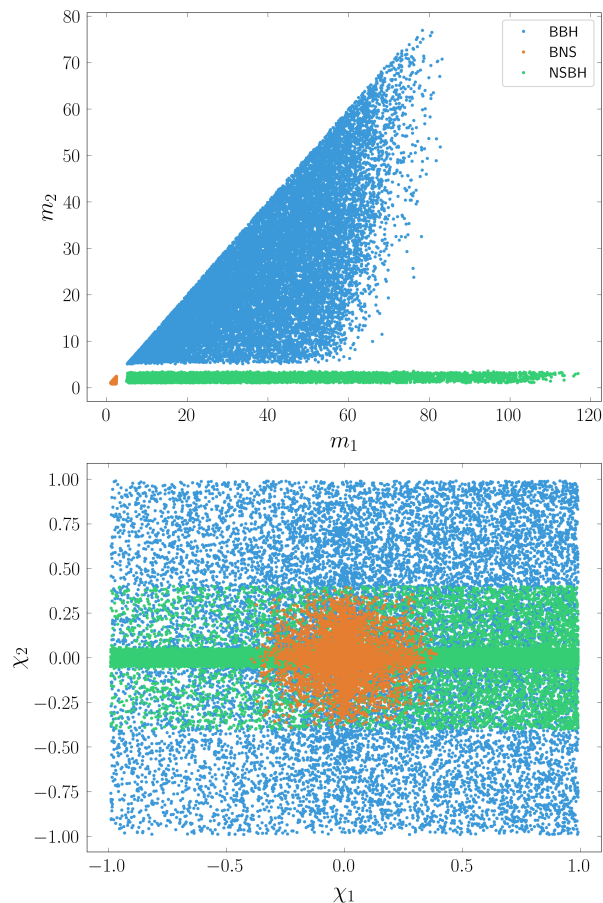


FIG. 1. The m_1 - m_2 (left) and χ_1 - χ_2 (right) parameter space of injections from the O2 MDC.

tems, respectively. Regression is applied to four features, namely the initial masses m_1 and m_2 and the initial spin magnitudes χ_1 and χ_2 . The training data has masses that span $m_1 = [1.01, 119]$ and $m_2 = [0.81, 81]$ and spin components ranging $\chi_{1,2} = [-0.99, 0.99]$, as shown in Fig. 1. For more information on the O2 MDC data mass and spin distributions, refer to Table I of [15].

ML algorithms require the data to be standardized, i.e. have a zero mean and unit variance. However, before standardizing the data we take an additional step to ensure astrophysical values from the predictions. In other words, a prediction without proper data conditioning can result in negative masses and spin magnitudes greater than one. To avoid this, we first map the data to be in the $(0, 1)$ range by solving a simple, linear system of equations separately for the masses and the spins. We, then, transform our data using a sigmoid function so that the values map between $(-\infty, +\infty)$. It is only at this point that the data is ready to be standardized and fed into the GPR and NN algorithms. Once the testing data predictions are completed, the reverse process is applied, i.e., the data is scaled back from standardization, exponentiated, and mapped back from $(0, 1)$ to astrophysical values.

III. METHODS

For a given problem, there are different machine learning algorithms that one can use to approximate a solution. For our purposes, we found Gaussian Process Regression and Neural Networks to be the most suitable methods due to their interpretability and accuracy, respectively.

A. Gaussian Process Regression

Gaussian process regression is a sophisticated mathematical tool to find joint Gaussian distributions between system data and the predictions. This makes it particularly powerful in interpolating between data sets and estimating errors. Throughout this paper, we use PyTorch's GPR module.

In ML algorithms one has to make certain choices to create a model. In the case of GPR, the kernel function, loss function, learning rate, and number of iterations need to be fixed. A kernel function is a covariance function used to measure the similarity between two data points. Our choice of kernel for training is the radial basis function (RBF) which is a squared exponential function given by

$$K(X_1, X_2) = e^{-\frac{\|X_1 - X_2\|^2}{2\sigma^2}}, \quad (1)$$

where X_1 and X_2 are input data points and σ is the variance, or lengthscale parameter. Apart from its popularity and versatility, we chose this kernel for its balance between accuracy and speed. Other kernels tested had the advantage of faster training but with a slightly higher loss in accuracy. More complex, slower kernels were not considered as there was no significant accuracy gain. We use the gradient-based Adams optimization algorithm [16] with a learning rate of 0.1 for 20 iterations to find the optimal hyperparameters. The loss function we use is the negative marginal log-likelihood. This combination of learning rate and iterations allows us to decrease the loss function at a quick rate without sacrificing accuracy.

B. Neural Networks (NNs)

While deep NNs have been extensively used in recent years for classification tasks, in this work we consider a NN for our regression analysis. We started by considering a NN implemented within the TensorFlow framework with Keras backend in order to have more flexibility in building the architecture of our NN. However we found that the best configurations between the ones tested is a standard NN with two hidden layers with 400 neurons each, ReLU activation function, and a linear output function. Such configuration can be easily coded with scikit-learn using the MLPRegressor. Since the scikit-learn is already imported in the LVK infrastructure, in the end we decide to use this framework. We train our NN for 10

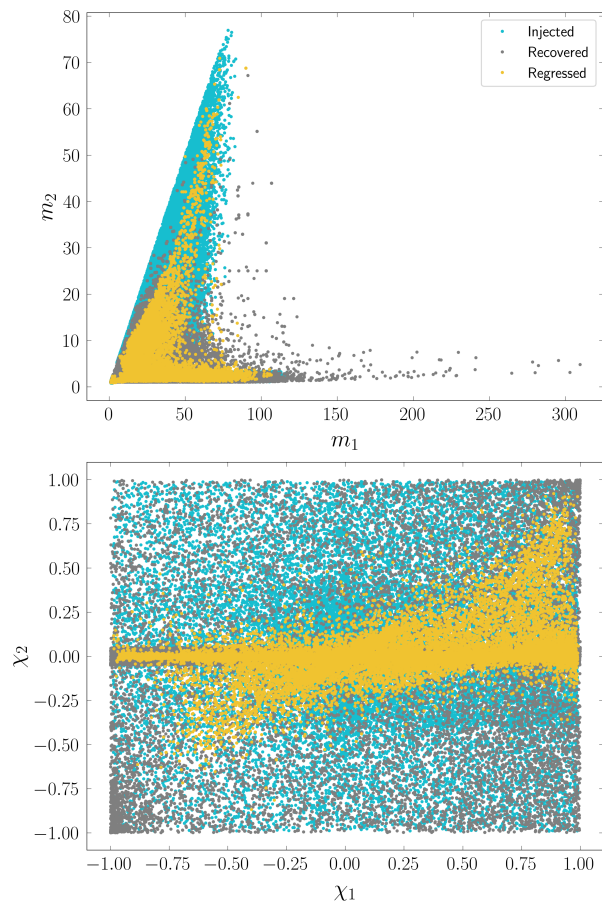


FIG. 2. The top panel shows the mass parameter space while the bottom panel shows the spin parameter space for the injected (teal), recovered (gray), and regressed (yellow) quantities. The regressed quantities shown here are using GPR.

epochs using a batch size of 128, a learning rate of 10^{-3} and mean square error (MSE) loss function.

IV. RESULTS

Using GPR and NNs to create a map between the injected and recovered masses and spins of the O2 dataset allows us to better estimate the intrinsic parameters of a binary system. Fig. 2 shows the mass (top panel) and spin (bottom panel) parameter space of the injections along with the recovered parameters from *GstLAL* and the regressed values. Although regression does not fully cover the parameter space spanned by the injections, it mimics the features well. For example, compared to the injected binaries, the recovered mass values tend towards a high m_1 and low m_2 , including several systems which are far from the expected mass-space. Instead, regression tries to correct for this tendency and includes the peak at high m_2 values without overestimating the m_1 range. The spin parameter-space is more complicated as most of the binaries include at least one non-spinning BH,

and therefore, there is less training data for most of the spin parameter space. However, the regression algorithm notices this feature and so it mainly populates that area of spin-space.

We further break down regression results by directly comparing the closeness of the recovered/regressed quantities to the injected ones. In Fig. 3, the gray circles represent the recovered values from the best-fit template, the orange circles are the predictions coming from GPR, and the blue circles are predictions from NN. The black lines shows where the points would lie if the parameters were perfectly recovered and/or perfectly regressed. This figure shows how the primary mass is sometimes recovered as having very high values. With regression, however, one can correct for these more extreme cases and predict values that are closer to the injected values. Opposite to this overestimation, there is the underestimation of the secondary mass. Regardless of the values of the secondary mass injected, the recovered value tends to be more often underestimated. Regression is able to counteract this bias and decrease the difference with the injected values. Apart from improving the mass estimates, one can see in the bottom left panels the effects regression has on spins. Although both the recovered and regressed values constitute wide bands around the injected values, the regressed values follow the diagonal line closer. In the secondary spin, the areas of parameter space furthest away from the true injected values are only populated by recovered values but not regressed values.

To get a better understanding on the distribution of errors for these four quantities, we show the mean relative error for the masses (top panels) and the mean difference for the spins (bottom panels). It is evident that the error distributions in all cases are more highly picked towards zero when using regression. In addition, we show the mean of the error distributions on the horizontal axis. Table I quantify this errors. Greater improvements are in the primary mass where errors drop from 35.2% to 12.7%. The secondary mass still sees a decrease in mean percent error by about 14.5%. The mean absolute differences in the spins also decrease by about half.

V. CONFIDENCE INTERVALS

In the previous section we have obtained two estimations from different ML algorithms. Now we discuss how we associate a confidence interval to this estimate and also to the *GstLAL* recovery. Since we want to compute confidence intervals for different prediction methods, we use a procedure that is algorithm-independent. The problem can be formulated as follows. Consider a vector of variables y . We apply a certain transformation \mathcal{N} so that we have a new set of variables $s = \mathcal{N}(y)$. Then we construct a new transformation \mathcal{M} that satisfies $x = \mathcal{M}(s) = \mathcal{M}(\mathcal{N}(y))$ where x is closest as possible to y . If \mathcal{N} is invertible, then the task is trivial since we can use $\mathcal{M} = \mathcal{N}^{-1}$ and thus $x = y$. However in general \mathcal{N} is not

TABLE I. Mean differences in absolute value $|\Delta\bar{y}| = \frac{1}{N}\Sigma|y_{\text{inj}}^i - y_{\text{rec/pred}}^i|$ and averages of the relative differences in absolute value $|\delta\bar{y}| = \frac{1}{N}\Sigma(|y_{\text{inj}}^i - y_{\text{rec/pred}}^i|/y_{\text{inj}}^i)$ for recovered and predicted data. The standard deviation $\sigma^{\Delta y}$ and $\sigma^{\delta y}$ are computed, respectively, from the difference and relative difference distributions without absolute value. Note that \mathcal{M}_c is not directly predicted but it is computed from the predicted masses m_i .

	$ \Delta\bar{y}_{\text{rec}} $	$\sigma_{\text{rec}}^{\Delta y}$	$ \delta\bar{y}_{\text{rec}} $	$\sigma_{\text{rec}}^{\delta y}$	$ \Delta\bar{y}_{\text{nn}} $	$\sigma_{\text{nn}}^{\Delta y}$	$ \delta\bar{y}_{\text{nn}} $	$\sigma_{\text{nn}}^{\delta y}$	$ \Delta\bar{y}_{\text{gpr}} $	$\sigma_{\text{gpr}}^{\Delta y}$	$ \delta\bar{y}_{\text{gpr}} $	$\sigma_{\text{gpr}}^{\delta y}$
m_1	6.625	12.052	0.352	0.596	3.456	6.976	0.134	0.292	3.241	...	0.127	0.279
m_2	2.761	7.178	0.256	0.351	1.457	3.583	0.123	0.331	1.414	...	0.111	0.319
χ_1	0.266	0.388	/	/	0.138	0.236	/	/	0.134	0.194	/	/
χ_2	0.277	0.460	/	/	0.153	0.268	/	/	0.151	0.225	/	/
\mathcal{M}_c	1.323	4.687	0.039	0.104	0.769	2.285	0.036	0.087	0.712	...	0.027	0.079

invertible and we have to rely on some heuristics to find \mathcal{M} , and the equality $x = y$ is not guaranteed. In other words, x is an estimate of y that we can obtain knowing $s = \mathcal{N}(y)$ and \mathcal{M} . In our specific case, y are the injected value, \mathcal{N} is the noise introduced by GstLAL, s are the recovered quantities, \mathcal{M} is a ML algorithm and thus x are the predictions of our ML algorithm. The requirement that x must be close to y can be formalized, for example, requiring that x minimizes a loss function. To compute the confidence intervals on the prediction x , we need to know the distribution of y values at fixed x , namely $f_x(y)$. We project $\mathcal{M}^i(s_1, s_2, \dots, s_i, \dots) = (x_1, x_2, \dots, x_i, \dots)$ on the (x^i, y^i) plane, so that for each value of x_i we can have different values of y_i and thus we can reconstruct $f_{x_i}(y_i)$. Treating each element of the arrays (features) separately means that we assume that the error associated to x^i can be determined knowing only the y^i distribution. Now the problem is reduced to how to compute the $f_{x_i}(y_i)$ distributions. We proceed to create a grid in the (x^i, y^i) plane and we populate it with the data from the training set. Then, we count how many points we have in each cell of the grid and we thus create a discrete density surface $\Sigma(x, y)$ that we save in an external file. When we want to compute the confidence interval, we then load the density surface $\Sigma(x, y)$, we interpolate the surface around the prediction x_p and the error-distribution is identified as $f_{x_p}(y) = \Sigma(x_p, y)$.

VI. CONCLUSIONS

Reiterate why what we did is important and how it improves current knowledge.

ACKNOWLEDGMENTS

We thank Deep Chatterjee and Shaon Ghosh for useful discussions and for sharing their work, which helped us compare our results to those of [15].

Part of this research was performed while the authors were visiting the Institute of Pure and Applied Mathematics (IPAM) at the University of California Los-Angeles

(UCLA). The authors would like to thank IPAM, UCLA, and the National Science Foundation through grant DMS-1925919 for their warm hospitality during the fall of 2021.

The authors are grateful for computational resources provided by the LIGO Laboratory and supported by the U.S. National Science Foundation Grants PHY-0757058 and PHY-0823459, as well as resources from the Gravitational Wave Open Science Center, a service of the LIGO Laboratory, the LIGO Scientific Collaboration and the Virgo Collaboration. L.M.Z. would like to express gratitude to the Mississippi Center for Supercomputing Research at the University of Mississippi for support in computing resources.

The work of L.M.Z. was partially supported by the MSSGC Graduate Research Fellowship, awarded through the NASA Cooperative Agreement 80NSSC20M0101.

All plots were made using the python package `matplotlib` [17].

This manuscript has been assigned LIGO Document Control Center number LIGO-P22XXXXX.

REFERENCES

- [1] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Observation of Gravitational Waves from a Binary Black Hole Merger, *Phys. Rev. Lett.* **116**, 061102 (2016), [arXiv:1602.03837 \[gr-qc\]](#).
- [2] R. Abbott *et al.* (LIGO Scientific, VIRGO, KAGRA), GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo During the Second Part of the Third Observing Run, (2021), [arXiv:2111.03606 \[gr-qc\]](#).
- [3] B. P. Abbott *et al.* (KAGRA, LIGO Scientific, Virgo, VIRGO), Prospects for observing and localizing gravitational-wave transients with Advanced LIGO, Advanced Virgo and KAGRA, *Living Rev. Rel.* **21**, 3 (2018), [arXiv:1304.0670 \[gr-qc\]](#).
- [4] G. Ashton and C. Talbot, Bilby-MCMC: an MCMC sampler for gravitational-wave inference, *Mon. Not. Roy. Astron. Soc.* **507**, 2037 (2021), [arXiv:2106.08730 \[gr-qc\]](#).
- [5] J. Wofford *et al.*, Expanding RIFT: Improving performance for GW parameter inference, (2022), [arXiv:2210.07912 \[gr-qc\]](#).
- [6] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, Bayesian parameter esti-

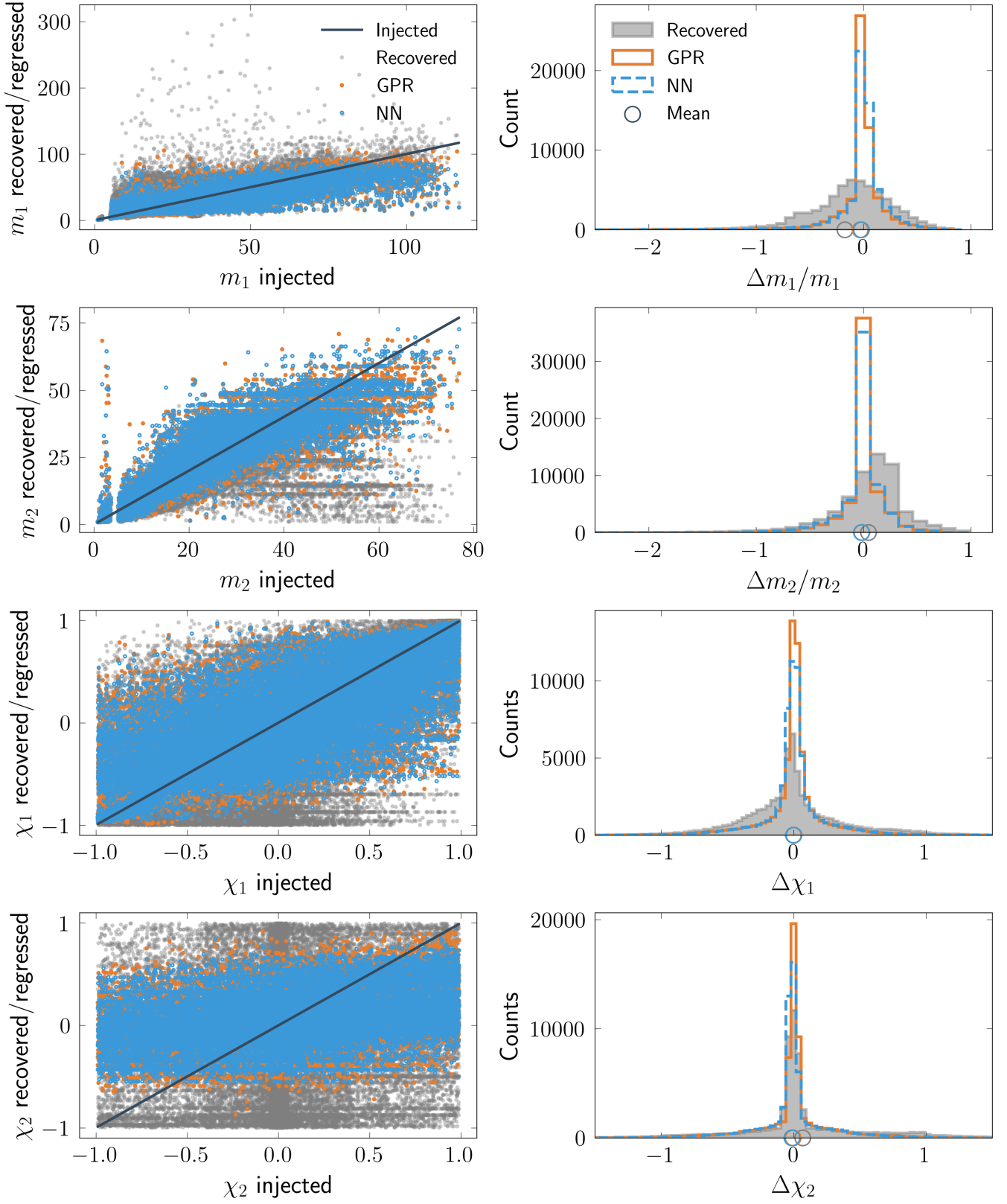


FIG. 3. The left side panels show the recovered (gray) and regressed masses and spins as functions of the injected values (solid line). Regressed values from GPR are in orange while those from NN are in blue. The right side panels show the mean error distributions.

- mation using conditional variational autoencoders for gravitational-wave astronomy, *Nature Phys.* **18**, 112 (2022), [arXiv:1909.06296 \[astro-ph.IM\]](#).
- [7] G. Ashton *et al.*, BILBY: A user-friendly Bayesian inference library for gravitational-wave astronomy, *Astrophys. J. Suppl.* **241**, 27 (2019), [arXiv:1811.02042 \[astro-ph.IM\]](#).
 - [8] J. Lange, R. O’Shaughnessy, and M. Rizzo, Rapid and accurate parameter inference for coalescing, precessing compact binaries, (2018), [arXiv:1805.10457 \[gr-qc\]](#).
 - [9] LIGO/Virgo Public Alerts User Guide, <https://emfollow.docs.ligo.org/userguide/>.
 - [10] H. Yu and R. X. Adhikari, Nonlinear Noise Cleaning in Gravitational-Wave Detectors With Convolutional Neural Networks, *Front. Artif. Intell.* **5**, 811563 (2022), [arXiv:2111.03295 \[astro-ph.IM\]](#).
 - [11] R. Ormiston, T. Nguyen, M. Coughlin, R. X. Adhikari, and E. Katsavounidis, Noise Reduction in Gravitational-wave Data via Deep Learning, *Phys. Rev. Res.* **2**, 033066 (2020), [arXiv:2005.06534 \[astro-ph.IM\]](#).
 - [12] C. J. Moore, C. P. L. Berry, A. J. K. Chua, and J. R. Gair, Improving gravitational-wave parameter estimation using Gaussian process regression, *Phys. Rev. D* **93**, 064001 (2016), [arXiv:1509.04066 \[gr-qc\]](#).
 - [13] D. Williams, I. S. Heng, J. Gair, J. A. Clark, and B. Khamesra, Precessing numerical relativity waveform surrogate model for binary black holes: A Gaussian process regression approach, *Phys. Rev. D* **101**, 063011 (2020), [arXiv:1903.09204 \[gr-qc\]](#).
 - [14] M. Walker, A. F. Agnew, J. Bidler, A. Lundgren, A. Macedo, D. Macleod, T. J. Massinger, O. Patane, and J. R. Smith, Identifying correlations between LIGO’s astronomical range and auxiliary sensors using lasso regression, *Class. Quant. Grav.* **35**, 225002 (2018), [arXiv:1807.02592 \[astro-ph.IM\]](#).
 - [15] D. Chatterjee, S. Ghosh, P. R. Brady, S. J. Kapadia, A. L. Miller, S. Nissanke, and F. Pannarale, A Machine Learning Based Source Property Inference for Compact Binary Mergers, *Astrophys. J.* **896**, 54 (2020), [arXiv:1911.00116 \[astro-ph.IM\]](#).
 - [16] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, (2014), [arXiv:1412.6980 \[cs.LG\]](#).
 - [17] J. D. Hunter, Matplotlib: A 2D Graphics Environment, *Comput. Sci. Eng.* **9**, 90 (2007).