Fast Bayesian gravitational wave parameter estimation using convolutional neural networks

M. Andrés-Carcasona [1]★ M. Martínez[1,2] and Ll. M. Mir[1]

[1]*Institut de Física d'Altes Energies (IFAE), The Barcelona Institute of Science and Technology, Campus UAB, E-08193 Bellaterra (Barcelona), Spain*
[2]*Catalan Institution for Research and Advanced Studies (ICREA), E-08010 Barcelona, Spain*

ABSTRACT

The determination of the physical parameters of gravitational wave events is a fundamental pillar in the analysis of the signals observed by the current ground-based interferometers. Typically, this is done using Bayesian inference approaches which, albeit very accurate, are very computationally expensive. We propose a convolutional neural network approach to perform this task. The convolutional neural network is trained using simulated signals injected in a Gaussian noise. We verify the correctness of the neural network's output distribution and compare its estimates with the posterior distributions obtained from traditional Bayesian inference methods for some real events. The results demonstrate the ability of the convolutional neural network to produce posterior distributions that are compatible with the traditional methods. Moreover, it achieves a remarkable inference speed, lowering by orders of magnitude the times of Bayesian inference methods, enabling real-time analysis of gravitational wave signals. Despite the observed reduced accuracy in the parameters, the neural network provides valuable initial indications of key parameters of the event such as the sky location, facilitating a multimessenger approach.

Key words: gravitational waves – software: data analysis.

1 INTRODUCTION

Gravitational waves (GW) are ripples in the fabric of space–time produced by moving massive objects such as binary black holes or rotating neutron stars. These waves were not directly detected until the recent discovery by the LIGO and Virgo collaborations in 2015 (Abbott et al. 2016). Since then, a total of 90 events have been detected in the three publicly released catalogues (Abbott et al. 2019, 2020, 2021a), corresponding to the O1–O3 observing runs of Advanced LIGO (Aasi et al. 2015) and Advanced Virgo (Acernese et al. 2015). The fourth observing run started in Spring 2023, already detecting tenths of new events. The majority of them are binary black hole mergers, but binary neutron star mergers and black hole-neutron star mergers have also been detected. The discovery of GWs has opened up a new window to study the Universe, as they can be used, for example, to infer the population of merging compact objects (Abbott et al. 2023b), test the theory of General Relativity (Abbott et al. 2021b) or understand the composition of neutron stars (Abbott et al. 2022a, b, c).

In order to extract useful information from the detected GW signals, it is necessary to accurately perform an estimation of the parameters of the source leading to the observed signal. Typically, this is done using a Bayesian inference framework, in which a probability distribution is constructed over the parameter space based on the observed data and prior knowledge (Veitch et al. 2015; Singer & Price 2016; Abbott et al. 2019; Ashton et al. 2019; Romero-

Shaw et al. 2020; Smith et al. 2020). Markov Chain Monte Carlo (MCMC) or Nested Sampling (NS) algorithms are commonly used to wander the parameter space and sample the posterior probability distribution (Gilks 2005; Skilling 2006; Veitch et al. 2015; Romero-Shaw et al. 2020). These methods, albeit powerful and precise, can be computationally intensive and do not scale appropriately with the increasing number of events. Due to the large volume of expected detections of future runs, in particular in next-generation experiments, new tools or implementations are being studied (Berry et al. 2015; Pankow et al. 2015; Singer & Price 2016; Cuoco et al. 2020; Alvey et al. 2023; Bhardwaj et al. 2023; Crisostomi et al. 2023).

Recently, there has been a growing interest in using artificial intelligence and machine learning techniques to address this task. Among those, convolutional neural networks (CNNs), normalizing flows, and autoencoders have been the most common approaches followed (Gabbard et al. 2018, 2022; George & Huerta 2018; Fan et al. 2019; Green et al. 2020; Green & Gair 2021; Krastev et al. 2021). CNNs have been applied to detect events (George & Huerta 2018; Menéndez-Vázquez et al. 2021; Morrás et al. 2022; Andrés-Carcasona et al. 2023), to perform the parameter estimation of the signals (Chua & Vallisneri 2020; Green et al. 2020; Gabbard et al. 2022) or to classify glitches (Zevin et al. 2017; George et al. 2018), among other applications (see Cuoco et al. 2020 for a comprehensive review).

The best performing machine learning algorithms applied to the parameter estimation problem are the ones described by Green et al. (2020), Green & Gair (2021), Gabbard et al. (2022), and Dax et al. (2023). Gabbard et al. (2022) use a variational autoencoder to

★ E-mail: mandres@ifae.es

Table 1. Parameter space of the training set. We additionally assume that $m_1 \geq m_2$.

Parameter	Symbol	Distribution	Units
Mass 1	m_1	Uniform(20, 60)	[$M_\odot$]
Mass 2	m_2	Uniform(20, 60)	[$M_\odot$]
Distance	d	Uniform(100, 3000)	[Mpc]
Right ascension	α	Uniform(0, 2π)	[rad]
Cosine of declination	$\cos(\delta)$	Uniform(-1, 1)	–
Polarization angle	ψ	Uniform(0, π)	[rad]
Inclination	θ_{JN}	Uniform(0, $\pi/2$)	[rad]
Time of coalescence	t_c	Uniform(0.5, 0.9)	[s]
Spin magnitude 1	a_1	Uniform(-1, 1)	–
Spin magnitude 2	a_2	Uniform(-1, 1)	–

predict the posterior probability distribution typically produced by a Bayesian inference approach. Their results are almost identical to those obtained using a Bayesian inference technique but to generate 8000 samples only takes 0.1s, an improvement of several orders of magnitude. A different approach is followed by Dax et al. (2023), where they estimate the Bayesian posterior using a neural network and then modify the distribution using importance sampling. This method takes between one and ten hours (depending on the waveform used) to perform the parameter estimation running on a GPU and multiple cores, which is a great improvement over the several hours to days that typically take the MCMC samplers to wander the parameter space. Finally, Green et al. (2020) and Green & Gair (2021) use normalizing flows to produce accurate posterior distributions, comparable to those produced by traditional methods.

In this paper, we present a new and fast method for GW parameter estimation using a CNN. This article is organized as follows. The training data set and pre-processing is explained in Section 2. In Section 3, the CNN architecture is presented. The training procedure is explained in Section 4. Finally, Section 5 shows the results obtained for the test set and for three real events, comparing it to the results obtained with the traditional MCMC approach.

2 DATA SET AND PRE-PROCESSING

To generate the training data for the CNN, we first take realizations of Gaussian noise that follow the power spectral density (PSD) of the strain measured for the O3b run in the three interferometers (Abbott et al. 2021a). Then, we simulate and inject on this noise GW signals using the IMRPhenomv2 waveform (Husa et al. 2016; Khan et al. 2016) from the PyCBC library (Usman et al. 2016; Nitz et al. 2020). The parameters of the injected signals are those described in Table 1. The delay between the different interferometers and its antenna patterns are also taken into account.

The strain is then whitened and normalized by dividing it by its standard deviation. We choose a sampling frequency of 4096 Hz and include the information of LIGO Hanford, LIGO Livingstone, and Virgo at the same time. The frequency is restricted to the (30 Hz, 1024 Hz) range, as the signals fit well within these values. The signal is then cropped to 1 s containing the merger between the time 0.5 s and 0.9 s (the exact coalescing time is taken as random over a uniform distribution). Therefore, the input matrix for the CNN has a shape of (4096, 3), each column corresponding to the data of each interferometer.

The signal-to-noise ratio (SNR) is defined as

$$\text{SNR} = \sqrt{4 \int_{f_{\min}}^{f_{\max}} df \frac{|\tilde{h}(f)|^2}{S_n(f)}}, \tag{1}$$

where $\tilde{h}(f)$ is the strain in frequency domain and $S_n(f)$ is the PSD of the strain noise. The network SNR is then computed as

$$\text{SNR}_{\text{net}} = \sqrt{\sum_{i=1}^{3} \text{SNR}_i^2}, \tag{2}$$

where i runs over the three interferometers. We restrict ourselves to signals that satisfy $\text{SNR}_{\text{net}} \geq 10$.

A total of 601 600 signals are used for the training stage, 38 400 for the validation, and 100 are kept for testing the performance afterwards.

An improved efficiency has been observed when fitting some derived quantities rather than the original ones presented in Table 1. For instance, instead of directly estimating the individual masses of the black holes, we utilize the chirp mass, denoted by $\mathcal{M}_c$ and which is calculated using the expression

$$\mathcal{M}_c = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}}, \tag{3}$$

alongside the mass ratio, denoted by q and computed as

$$q = \frac{m_1}{m_2}, \tag{4}$$

where $m_1 \geq m_2$.

Similarly, instead of predicting the individual spins, we use the effective spin, defined as

$$\chi_{\text{eff}} = \frac{a_1 m_1 + a_2 m_2}{m_1 + m_2}. \tag{5}$$

Furthermore, we make the decision not to fit the polarization and inclination angles, as they are not among the key parameters required for real-time analyses. As a result, the CNN will estimate the set of parameters $\boldsymbol{\theta} = \{\mathcal{M}_c, q, d, t_c, \chi_{\text{eff}}, \alpha, \delta\}$.

3 CNN ARCHITECTURE

The CNN architecture that has been used in this work is displayed in Fig. 1. The first set of layers are one-dimensional convolutions and one-dimensional max pooling ones, followed by a set of fully connected dense layers. Between each of the dense layers, a dropout 20 per cent is applied to prevent overfitting during the training stage.

To be able to capture the uncertainty in the final estimation of the parameters, we choose to model the full distribution as the posterior in Bayesian statistics would, instead of producing point estimates. Therefore, the last dense layer that contains 14 neurons will be connected to a TensorFlow Probability layer with a Kumaraswamy distribution for each parameter.

The Kumaraswamy distribution has a probability density function defined by (Kumaraswamy 1980)

$$f(x; \alpha, \beta) = \alpha \beta x^{\alpha-1} (1 - x^\alpha)^{\beta-1}, \tag{6}$$

for $x \in [0, 1]$, which is a flexible enough distribution. It is very similar to the beta distribution, but its simple expression of the probability density function allows to have an easy evaluation of the quantile function, which takes the form of

$$F^{-1}(\xi; \alpha, \beta) = (1 - (1 - \xi)^{1/\beta})^{1/\alpha}. \tag{7}$$

Sampling from this distribution is then trivial by applying the inverse transform sampling method. If we draw $\xi \sim \text{Uniform}(0, 1)$ and then compute $F^{-1}(\xi; \alpha, \beta)$ for each realization, we will obtain the desired samples of this distribution. This procedure is very fast in contrast to sampling from a distribution for which no
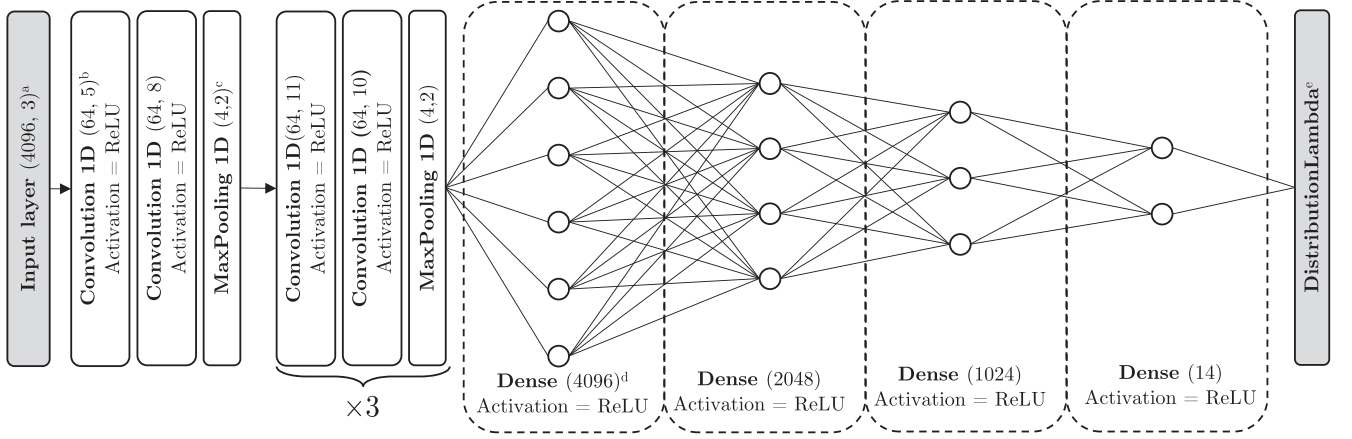
Figure 1. Architecture of the CNN used in this work. [a] Size of the input layer. [b] Filters and size of the kernel of the 1D convolution layer. [c] Size of the pool size of the 1D maximum pooling layer. [d] Number of neurons of the fully connected dense layer. [e] Layer that includes the 7 Kumaraswamy distribution (one per each estimated parameter). Before this layer, the exponential of the numbers outputted by the last dense fully connected layer are taken.
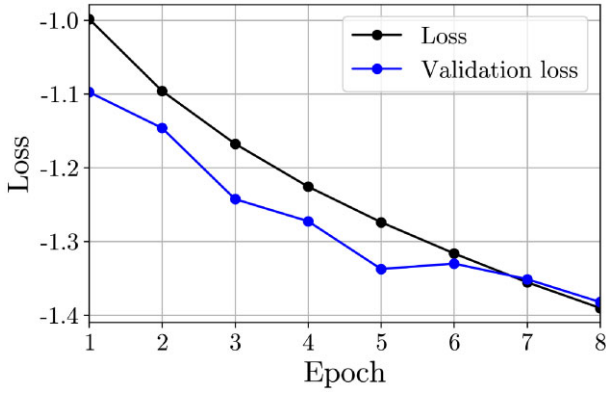

Figure 2. Metrics tracked during the training procedure.
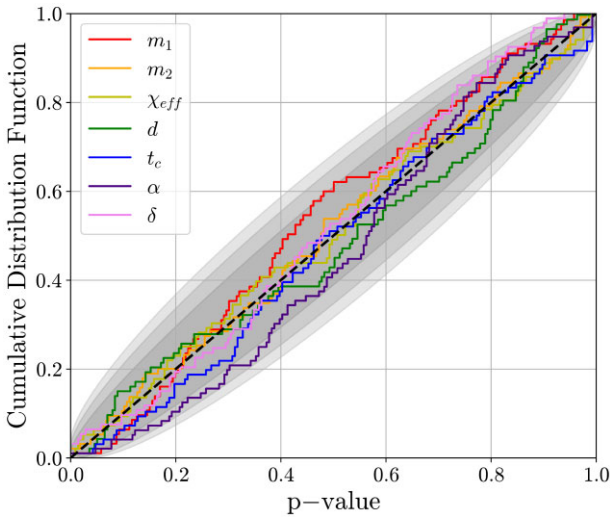
Figure 3. P–P plot for the different variables that are being fitted by the CNN. The individual masses are obtained by post-processing the chirp mass and mass ratio. The shaded regions indicate the $\pm 1\sigma$, $\pm 2\sigma$, and $\pm 3\sigma$ confidence levels in decreasing intensity of colour, respectively.

analytical expression of the quantile function is available or which is computationally expensive to evaluate.

Since our variables are not in the range [0, 1], the estimated parameters need to be transformed as

$$x = \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}} , \tag{8}$$

where $\theta \in \boldsymbol{\theta}$.

The numbers estimated for the different variables are different in orders of magnitude and, therefore, we estimate the $\log \alpha$ and $\log \beta$ instead. This motivates taking the exponential of the 14 numbers generated by the last dense layer before evaluating the probability distribution.

The loss function that will be used is the negative log-likelihood (i.e. $L = -\log \mathcal{L}(\alpha, \beta | X)$). For the Kumaraswamy distribution and n observations, the log-likelihood equals

$$\log \mathcal{L} = n \log \alpha + n \log \beta + (\alpha - 1) \sum_{i=1}^{n} \log(x_i)$$
$$+ (\beta - 1) \sum_{i=1}^{n} \log(1 - x_i^{\alpha}) . \tag{9}$$

Choosing this loss function is equivalent to maximizing the log-likelihood, meaning that the neural network is actually learning to find the parameters of the distribution that best fit the data.

4 TRAINING

To build and train the CNN, we use Keras with TensorFlow's backend and its implementation in GPUs (Abadi et al. 2016). For managing the probabilistic layers, we use distribution layers (Dillon et al. 2017) from TensorFlow Probability.

We feed the data in batches of 32 signals as was done in Menéndez-Vázquez et al. (2021) and Andrés-Carcasona et al. (2023). The learning rate is set to 10^{-5}. The metrics tracked during this stage are mainly the loss and the validation loss. The former is computed with the training set and represents the function being minimized, while the latter is computed using the validation set. This allows to observe the appearance of overfitting or underfitting during the training procedure. The training lasts for 8 epochs and the one yielding a minimum validation loss is going to be the neural network used for inference.
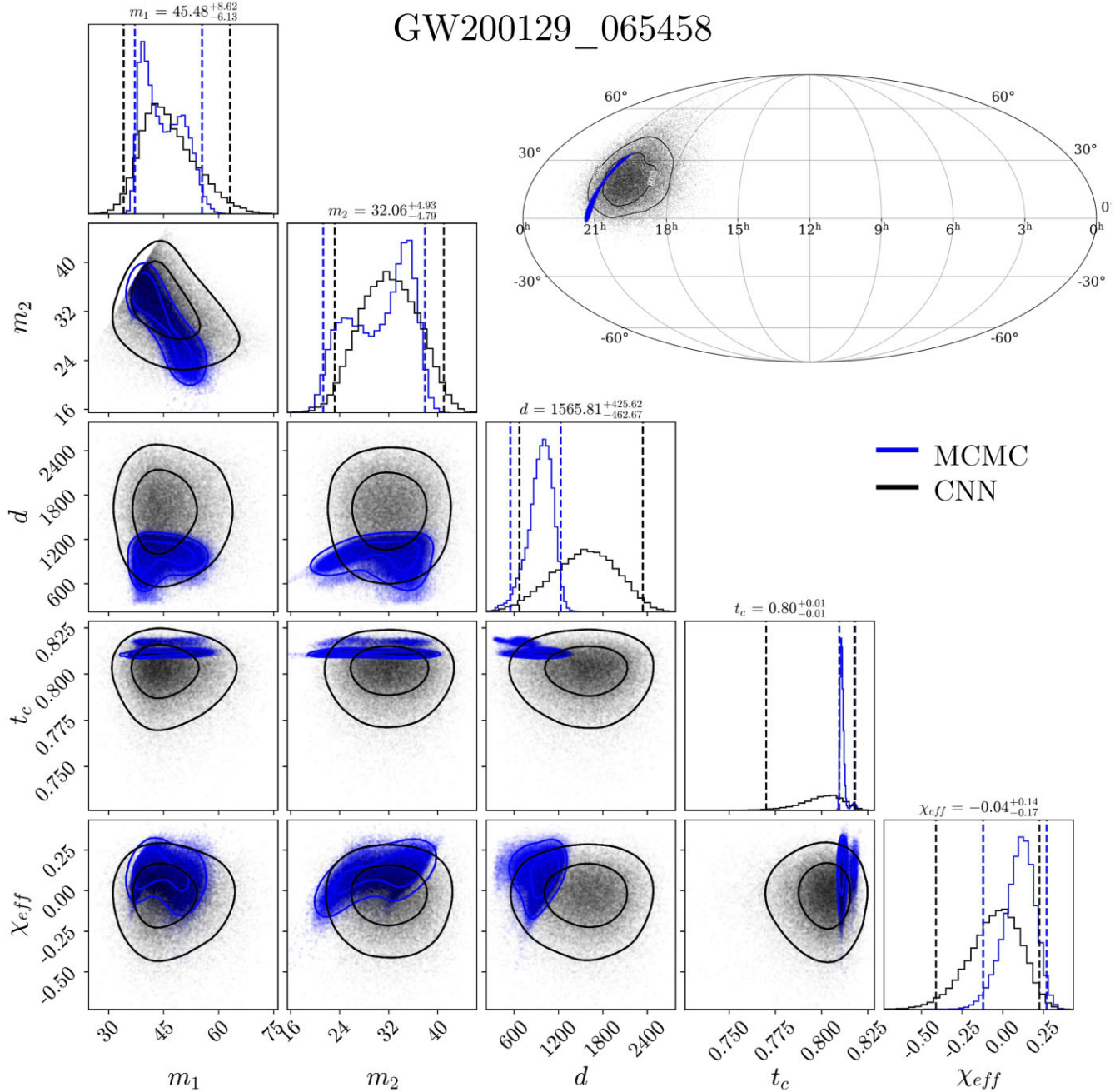
Figure 4. Full posterior distribution obtained for the CNN and for the public parameter estimation release obtained with a MCMC approach for the GW200129_065458 event.

The evolution of the metrics tracked is displayed in Fig. 2. The loss, as expected, decreases steadily, indicating that the CNN is continuously learning to fit the data. This indicates a good behaviour over the training set. The validation loss also displays a decrease during the training indicating that the CNN is also learning how to generalize the results over data that has not seen before.

The final loss and validation loss that are obtained for the CNN chosen, the one of the eighth epoch, are both -1.39. This CNN is tested with the other data set and two real events in the following section.

It is worth noting that the CNN performance could be further improved by expanding the training set to include a wider range of

signals and noise conditions. Moreover, a higher fine-tuning of the model's hyperparameters, such as adjusting the learning rate or the architecture, could potentially enhance its accuracy and reduce a bit more the uncertainties in the estimated parameters. This is out of the scope of the paper, as it intends to be a proof of concept.

5 RESULTS

In this final section, the main results of the CNN are presented. The first test that can be performed to ensure the proper performance is constructing the probability–probability (or P–P for short) plot. This plot is constructed by performing the inference of the CNN over
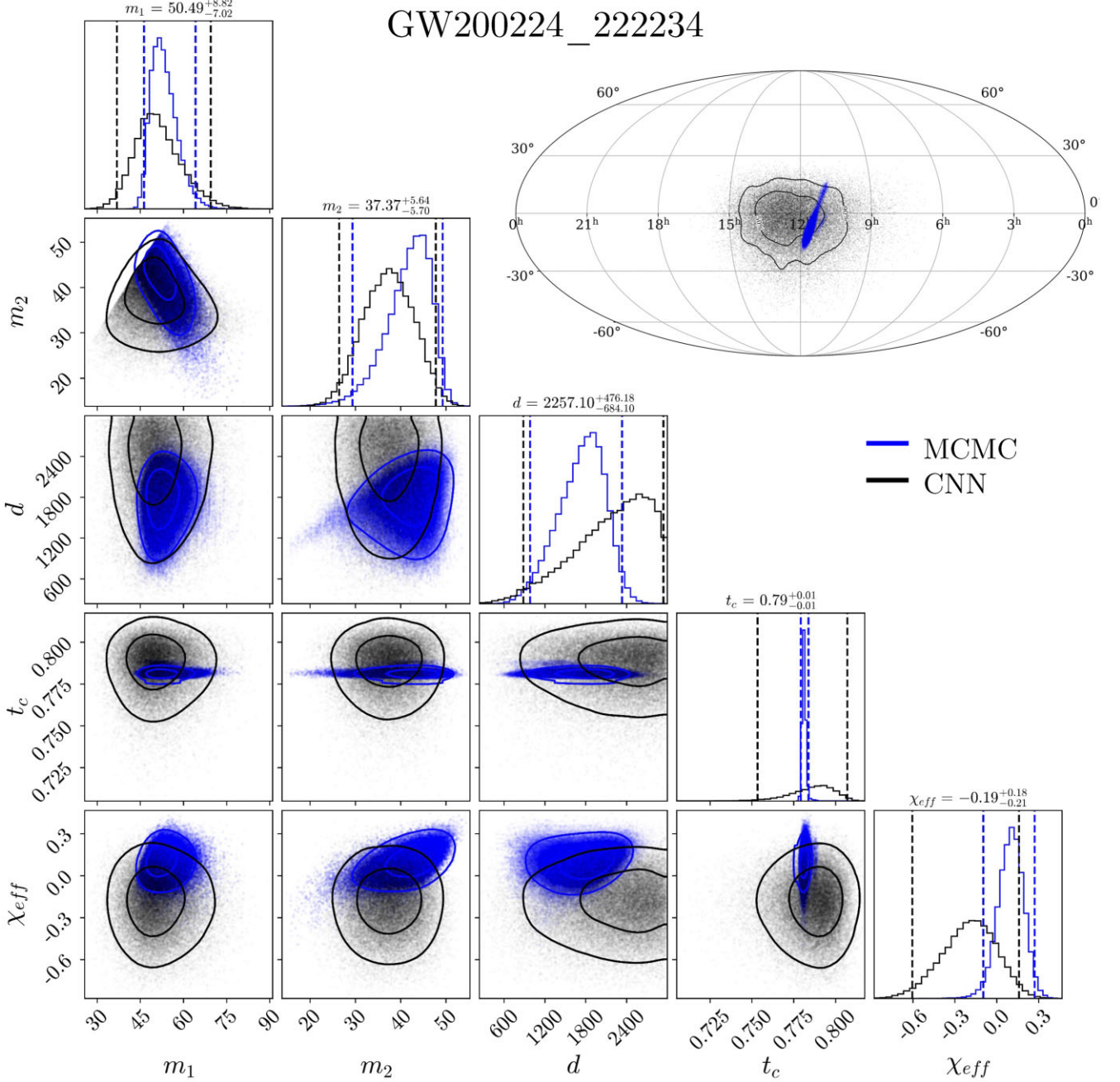
Figure 5. Full posterior distribution obtained for the CNN and for the public parameter estimation release obtained with a MCMC approach for the GW200224_222234 event.

the test set and evaluating the *p*-value of the true parameter. This is computed as

$$p\text{-value} = \int_{\theta_{\text{true}}}^{\infty} f(\theta; \alpha, \beta) \, d\theta \ . \tag{10}$$

If the distribution outputted by the CNN is a true statistical distribution, the cumulative probability density of this *p*-value should follow a 45° line between the points (0, 0) and (1, 1). This plot is shown in Fig. 3.

These cross-validation results demonstrate the robustness of our CNN approach, as it consistently produces a reliable parameter estimation using a new subset of data.

Since the CNN has been trained on simulated signals and on Gaussian noise, it is interesting to test it using real data. The GWTC-3 catalogue (Abbott et al. 2021a) contains the 35 O3b confident detections. From this collection, a judicious selection of events is needed following certain criteria. Namely, the chosen events must have been detected during periods when all three interferometers were actively online and when the data quality recorded across these interferometers was designated as optimal. Furthermore, compatibility with the CNN's training regimen requires that the parameters derived through conventional methodologies lie within the pre-determined range calibrated for the neural network. Finally, only events exhibiting a SNR exceeding 10 are considered viable candidates as this was
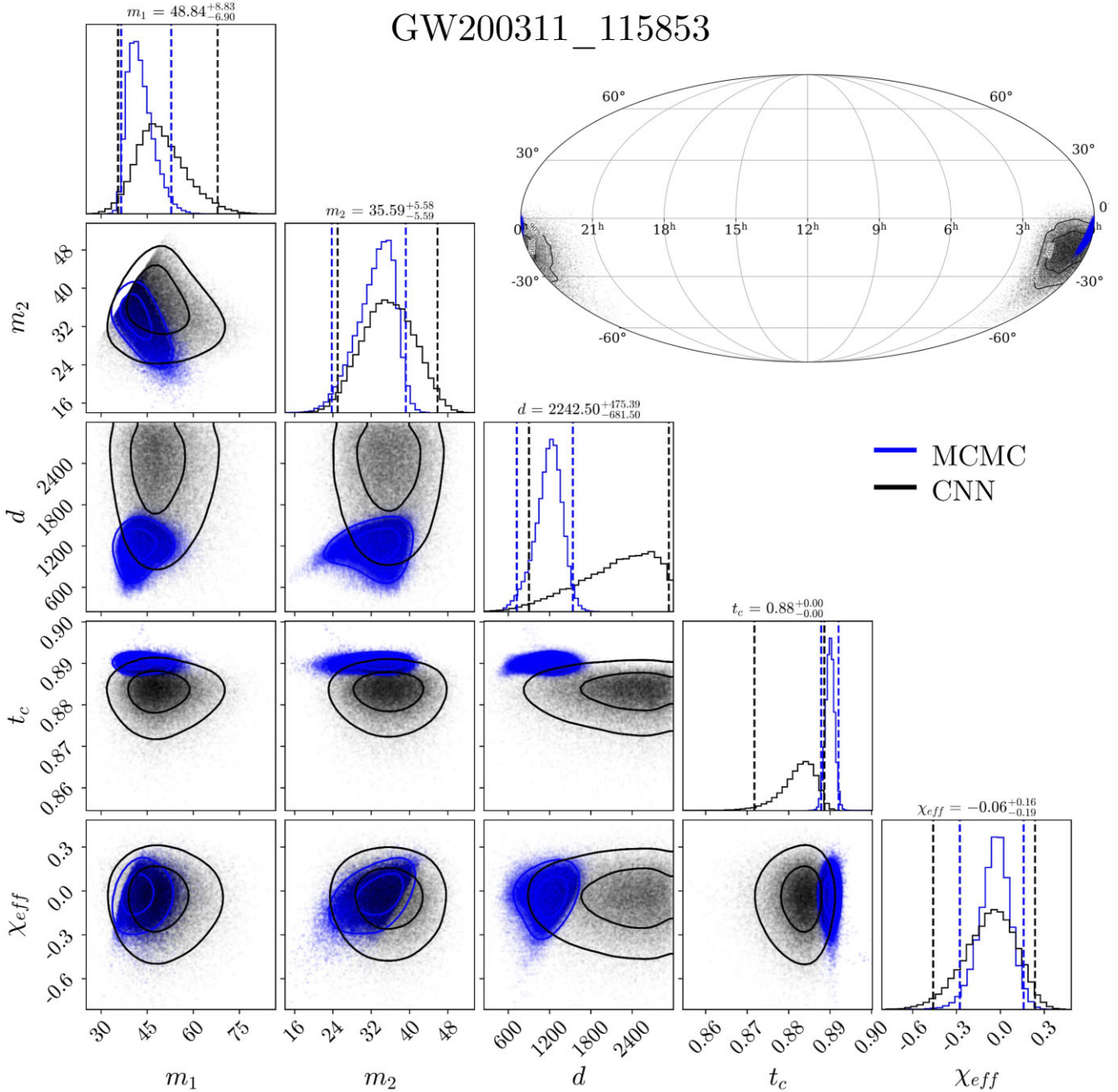
Figure 6. Full posterior distribution obtained for the CNN and for the public parameter estimation release obtained with a MCMC approach for the GW200311_115853 event.

a decision during training. Accordingly, among these stringent criteria, the events GW200129_065458, GW200224_222234, and GW200311_115853 emerge as good choices for this validation, fulfilling the requirements and having different parameters.

For GW200129_065458, we apply the CNN on the data that contains it and compare it to the posterior distributions published in the GWTC-3 catalogue by the LVK collaboration using the traditional Bayesian inference methods (Abbott et al. 2021a, 2023a). The result is shown in Fig. 4. To generate 10 000 samples of the posterior the CNN, running on a GPU NVIDIA GeForce RTX 2080 Ti, takes 0.05 s. This implies an improvement of several orders of magnitude with respect to the traditional method.

Our CNN is able to produce posterior distributions that in all cases are compatible with the published results. The one that exhibits the worst similarity is, for this particular event, the distance, as it overestimates it. In this case, the sky position has a big overlap with the MCMC estimated one, but still has a larger uncertainty.

The results for the event GW200224_222234 are shown in Fig. 5. In this case, all the estimated parameters are also in accordance with those published in the GWTC-3 catalogue. The one that exhibits the worst performance is, in this case, the effective spin. Regarding the sky position, the uncertainty yielded by the CNN is too large to accurately pinpoint the event, but it still is unbiased and could produce a first alert for the instruments that look for electromagnetic

counterparts to start pointing their instruments to a given patch in the sky while the more accurate pipelines improve the precision of the sky localization.

A typical short gamma ray burst, as the one observed alongside GW170817 (Abbott et al. 2017), lasts less than 2 s and, taking into account that the instruments might need time to point towards a given direction, ideally the inference speed should be of a fraction of a second to avoid becoming the bottleneck. This is achieved by our neural network.

Finally, the results for the GW200311_115853 event are shown in Fig. 6. In this particular example, the coalesce time is slightly underestimated and the distance overestimated. The rest of the parameters are well predicted but with larger uncertainties.

Overall, the results indicate a good response from CNN. An important advantage of our approach is its computational efficiency. The CNN model provides several posterior samples in a fraction of the time required by Bayesian inference methods. This enhanced capability enables real-time analysis of GW signals and even facilitates prompt alerts for possible follow-up observations and multimessenger astronomy collaborations.

Despite the high inference speed, the CNN approach introduces a trade-off in terms of higher parameter uncertainty. These can be mainly attributed to the simplified assumptions made by the CNN architecture and the limited training data used. However, even with these uncertainties, the CNN model can serve as an initial indication for the sky position and other key parameters, enabling a swift response and triggering more accurate analyses by other slower, yet more accurate, pipelines.

In summary, with these results, our CNN model demonstrates promising capabilities for a fast parameter estimation of GWs. It provides a reliable posterior distribution, while exhibits a competitive performance compared to traditional Bayesian sampling methods, and offers real-time inference capabilities. This is only a proof of concept and future research can focus on refining the CNN architecture, incorporating additional data sources, and exploring ensembling techniques to further improve the accuracy and robustness of GW parameter estimation.

6 CONCLUSIONS

We have presented a neural network to perform the typically computationally expensive task of estimating the parameters of GW events. The chosen architecture has been a one-dimensional CNN with three channels, one per available interferometer, and predicting the parameters of a Kumaraswamy distribution. These parameters then define what can be regarded as the posterior distribution. The training shows that this architecture can correctly learn how to estimate them and that the outputted distribution behaves like a probability density function. Finally, it has been applied to real events, and the results have shown that, although our CNN has a larger uncertainty than traditional Bayesian inference approaches, the actual parameters are generally in agreement with the outputted distribution. Our approach offers real-time inference speeds that could be used for possible multimessenger follow-ups.

ACKNOWLEDGEMENTS

DATA AVAILABILITY

The codes and data generated for this paper are available under reasonable request to the atuhors.

REFERENCES

Aasi J. et al., 2015, Class. Quantum Gravity, 32, 074001
Abadi M. et al., 2016, in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, p. 265
Abbott B. et al., 2016, Phys. Rev. Lett., 116, 061102
Abbott B. P. et al., 2017, Phys. Rev. Lett., 119, 161101
Abbott R. et al., 2019, Phys. Rev. X, 9, 031040
Abbott R. et al., 2021, Phys. Rev. X, 11, 021053
Abbott R. et al., 2021a, preprint (arXiv:2111.03606)
Abbott R. et al., 2021b, preprint (arXiv:2112.06861)
Abbott R. et al., 2022a, Phys. Rev. D, 106, 042003
Abbott R. et al., 2022b, Phys. Rev. D, 106, 102008
Abbott R. et al., 2022c, ApJS, 935, 1
Abbott R. et al., 2023a, ApJS, 267, 29
Abbott R. et al., 2023b, Phys. Rev. X, 13, 011048
Acernese F. et al., 2015, Class. Quantum Gravity, 32, 024001
Alvey J. , Bhardwaj U., Nissanke S., Weniger C., 2023, preprint (arXiv:2308.06318)
Andrés-Carcasona M. , Menéndez-Vázquez A., Martínez M., Mir Ll. M., 2023, Phys. Rev. D, 107, 082003
Ashton G. et al., 2019, ApJS, 241, 27
Berry C. P. L. et al., 2015, ApJ, 804, 114
Bhardwaj U. , Alvey J., Miller B. K., Nissanke S., Weniger C., 2023, Phys. Rev. D, 108, 042004
Chua A. J. K., Vallisneri M., 2020, Phys. Rev. Lett., 124, 041102
Crisostomi M. , Dey K., Barausse E., Trotta R, 2023, Phys. Rev. D, 108, 044029
Cuoco E. et al., 2020, Mach. Learn.: Sci. Technol., 2, 011002
Dax M. , Green S. R., Gair J., Pürrer M., Wildberger J., Macke J. H., Buonanno A., Schölkopf B., 2023, Phys. Rev. Lett., 130, 171403
Dillon J. V. et al., 2017, preprint (arXiv:1711.10604)
Fan X. , Li J., Li X., Zhong Y., Cao J., 2019, Sci. China Phys. Mech. Astron., 62, 969512

Foreman-Mackey D., 2016, J. Open Source Softw., 1, 24

Gabbard H. , Williams M., Hayes F., Messenger C., 2018, Phys. Rev. Lett., 120, 141103

Gabbard H. , Messenger C., Heng I. S., Tonolini F., Murray-Smith R., 2022, Nat. Phys., 18, 112

George D., Huerta E. A., 2018, Phys. Lett. B, 778, 64

George D. , Shen H., Huerta E. A., 2018, Phys. Rev. D, 97, 101501

Gilks W. R., 2005, Markov Chain Monte Carlo, Encyclopedia of Biostatistics. John Wiley and Sons, Ltd. Chichester, UK

Green S. R., Gair J., 2021, Mach. Learn.: Sci. Technol., 2, 03LT01

Green S. R. et al., 2020, Phys. Rev. D, 102, 104057

Husa S. , Khan S., Hannam M., Pürrer M., Ohme F, Forteza X. J., Bohé A., 2016, Phys. Rev. D, 93, 044006

Khan S. et al., 2016, Phys. Rev. D, 93, 044007

Krastev P. G., Gill K., Villar V. A., Berger E., 2021, Phys. Lett. B, 815, 136161

Kumaraswamy P., 1980, J. Hydrol., 46, 79

Menéndez-Vázquez A. , Kolstein M., Martínez M., Mir Ll. M., 2021, Phys. Rev. D, 103, 062004

Morrás G. , García-Bellido J., Nesseris S., 2022, Phys. Dark Universe, 35, 100932

Nitz A. et al., (2020), gwastro/pycbc: PyCBC release v1. 16.11, https://doi.org/10.5281/zenodo.4134752 Last access: 24 Nov 2023

Pankow C. , Brady P., Ochsner E., O'Shaughnessy R., 2015, Phys. Rev. D, 92, 023002

Romero-Shaw I. M. et al., 2020, MNRS, 499, 3295

Singer L. P., Price L. R., 2016, Phys. Rev. D, 93, 024013

Skilling J., 2006, Bayesian Anal., 1, 833

Smith R. J. E., Ashton G., Vajpeyi A., Talbot C., 2020, MNRAS, 498, 4492

Usman S. A. et al., 2016, Class. Quantum Gravity, 33, 215004

Veitch J. et al., 2015, Phys. Rev. D, 91, 042003

Zevin M. et al., 2017, Class. Quantum Gravity, 34, 064003

This paper has been typeset from a TEX/LATEX file prepared by the author.