Gravitational wave search with Machine Learning

Author: Roberto Bada Nerín.
*Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.**

Advisor: Oleg Bulashenko

Abstract: We train two Deep Learning (DL) classifiers, based on VGG19, using Gravitational Waves (GW) simulated data. The first of them (from now on, N-S) is capable of distinguishing between plain noise and a simulated signal with injected noise. The second one (from now on, L-U) is trained with noisy GW signals and is able to tell if said signals are lensed GW or not. For our simulations, we consider Binary Black Holes (BBH) systems with spinless members that have masses between 10 and 80 solar masses. The luminosity distances detector-source lie between 500 and 3500 Mpc and we discard events with Signal-to-Noise Ratio (SNR) smaller than 5 or bigger than 50. We feed our models with images of the Q transform of these strains of data, finding that VGG19 performs well in both classifications: when classifying the test sets, N-S achieved an accuracy of 100%, while L-U achieved an accuracy of 98%. The conclusion of this work is that it ratifies the potential GW has, not only in the detection of GW signals, but also in the study of other predicted effects such as lensing.

I. INTRODUCTION

Since the first detection of a GW on 14 September 2015, the LIGO-Virgo Collaboration has detected more than 90 events [1, 2]. The beginning of a fourth observing run this past month of May (O4), the advances in noise-reduction and the different projects concerning new detectors (LISA, LIGO India, etc.) demonstrate the importance that GW search is going to adopt in Astronomy during the following years. The recollection of big amounts of data will facilitate the study of things like the equation of state of neutron stars or the Hubble tension, while being a tool for testing General Relativity (GR) predictions, such as lensing (GR predicts that GWs can be lensed by big amounts of mass, just like light does). To this day, none of the more than 90 events has been proved to be a lensed GW.

In this context of data recollecting, we find ourselves in the need of technology capable of analyzing this data in a fast and trustworthy way. For this task, Machine Learning (ML) has been proved to be a good tool. The combination of ML techniques, together with the traditional GW search methods (Matched Filtering (MF) [3] for detection and Bayesian methods for parameter estimation), could save researchers time, while retaining the same levels of accuracy.

There have been many previous studies that have considered the use of ML for GW search. In [4], Kim et al. improved the efficiency of traditional statistical methods for detection by using an artificial neural network, and suggested to combine these methods. The cited article was received and revised in 2015 before the first detection of a GW. In [5], Jiang, Yang and Li were able to detect all 41 events from O1, O2 and O3a using a Convolutional Neural Network (CNN). Furthermore, this network was able to analyze 4096 seconds of data in 21 minutes, and the authors achieved a False Alarm Ratio (FAR) of one false positive in a two months period. In [6] and [7], Kim et al. applied VGG (a CNN) for the search of lensing phenomena in detected GW events. Although these models achieved high accuracy when tested, they did not find lensed GW among the studied events. In [8], Jin et al. used the 2D-UNet algorithm, a CNN created in the context of medical image recognition, to detect GWs with accuracy. They argued that this neural network could even find prior masses for parameter estimation, which would save significant time in this task. In adidition to event detection and parameter estimation, ML has proven useful for other GW-related topics. A good example is Gravity Spy [9], a ML model capable of classifying more than 20 different types of glitches.

The objective of this project is to draw a first approach to the use of ML in GW search by training VGG19, a well-known CNN, so that it can distinguish different images of simulated data in the time-frequency domain into the categories Lensed Signal, Unlensed Signal or Noise. Section II is dedicated to explaining the procedure for obtaining the simulated data, the ranges of parameters used and the shape of the different datasets. Some information about the type of data we work with in GW search will be given. In section III, the reader will find all the necessary information concerning the trained models, from the structure of these to the training process. Finally, section IV presents the results after the testing phase and section V states the conclusions of this dissertation.

II. THE DATA

The usual procedure would be training our models with real data, but nowadays we have only detected around 90 events. Luckily, GW data can be easily simulated, and simulated data of GWs has been used before in ML mod-

*Electronic address: rbadan99@gmail.com

els with excellent results. But before going into detail, it is necessary to talk about the different data we use when studying GWs. The interferometers measure the differential change of the length of the arms of the detector, giving us a temporal series of the strain of the signal $h(t)$ that looks like FIG. 1:

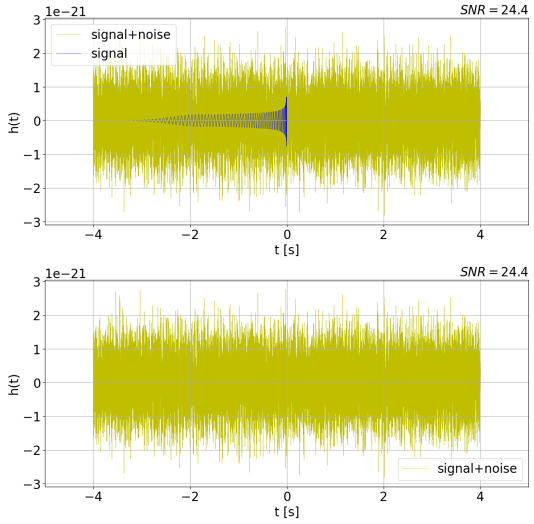

FIG. 1: Time series of a simulated GW with injected noise. In blue we can see the hidden signal (upper image). The second image shows the whole strain, without highlighting the signal.

As this figure shows, it would be impossible to find the signal in the time series. This is much easier to spot in the time-frequency domain, as it is shown in FIG 2:

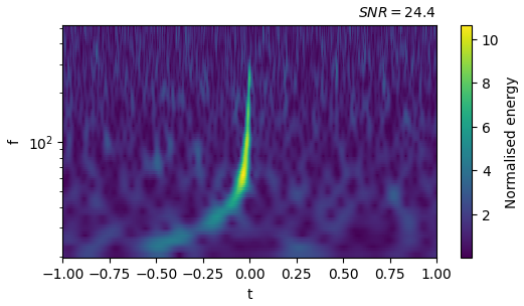

FIG. 2: Q-transform of the time series in FIG. 1. Notice how the signal has become visible.

FIG. 2 is obtained from the strain data used to plot FIG. 1 by applying what is called a Q-transform, a type of time-frequency transform that has some special advantages in front of other types of transforms [10]. The reader can find more details about the Q-transform in [11]. For now, it is enough to say that this transform has proven to be very useful in GW search.

Since FIG. 2 gives us a more visual representation than FIG. 1, we choose this type of data for training our CNNs. The reason is that these networks are specialized for finding differences between images.

A. Noise

We generate more than 20,000 images of Q-transforms of strains of data without signals hidden. Instead of using white noise, we use colored noise, which is Gaussian noise with a frequency-dependent variance; white noise, on the other hand, has frequency-independent variance. This makes our data more realistic. In order to give the model noise templates as realistic as possible, we use the Power Spectral Density (PSD) from three detectors: the two LIGO detectors (Handford and Livingstone) and the Virgo detector (Pisa). These densities are well-known and can be found in different Python libraries such as PyCBC [12]. They model the probability of finding noise at each frequency. The images have a duration of 2 seconds, and the PSDs from the three mentioned detectors are used randomly when generating each image. Once we have all the images, we divide them in three sets: around 18,000 will be used as training set (the images the model will use for training), 1,000 as validation set (these images help us controlling some issues that could happen during training such as over-fitting, a phenomenon that takes place when the model performs good when classifying the training set, but is unable to generalize) and 1,000 as test set (this set will be used to evaluate the model performance over images that it has never seen).

B. Unlensed Signals

We generate around 10,000 images of Q-transforms of BBH mergers in which noise from the mentioned PSDs has been injected. The black holes considered are spinless and they are taken to be perfectly oriented with respect to the source. We use the IMRPhenomPv2 templates from PyCBC library with no spin or precession included. The masses are randomly generated following a log-uniform distribution between 10 and 80 solar masses. For each pair of masses, we generate 10 different images with different luminosity distance. This distance lies in the set $\{500, 750, 1000, 1250, 1500, 1750, 2000, 2500, 3000, 3500\}$ (in Mpc). All simulated events have SNR between 5 and 50. We use around 9,000 images for training, 500 for validation and 500 for testing.

C. Lensed Signals

Just as waves of light undergo lensing when interacting with a convex-shaped material, gravitational waves can experience a similar phenomena. In practice, this could be found in magnified signals, multiple images of the same signal and other patterns that could be distinguished by ML models [6, 7]. We generate around 10,000 images of Q-transforms of lensed BBH mergers with injected noise. Here, we consider the microlensing case, for which two virtual images interfere. These images have

the same characteristics than the simulated unlensed signals, with the addition of two parameters: a position parameter y that relates the distance to the source and to the lens, and that is randomized uniformly between 0 and 1, and the time delay Δt, related to the mass of the lens [13]. This last parameter is randomized uniformly between 0.225 ms and 0.5 s, and it tells us the delay between the two signals received consequence of the lensing process. We use around 9,000 images for training, 500 for validation and 500 for testing.

Note: All the ranges of parameters mentioned, and the distributions used, have been taken from the different references involving training of ML models, adapting them to the characteristics of our data and models. Also, since images are taken or deleted depending on their SNR, the size of the training, validation and test sets vary. This is not a problem since, although this number is not the same, in each set, the number of images of each class is compensated.

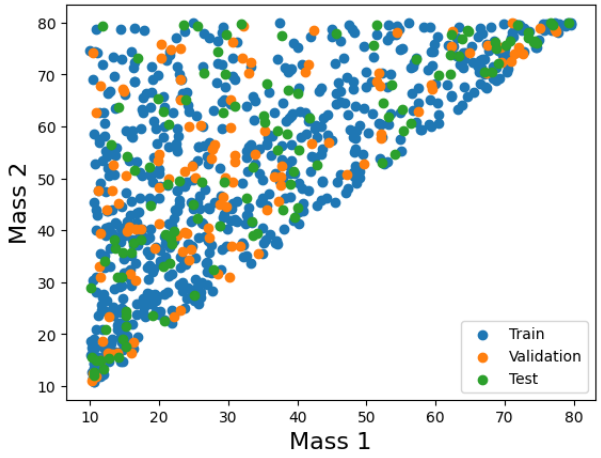

FIG. 3: Distribution of masses for the lensed and unlensed signals, differentiated by the set they belong to. Notice that, since we used a log-uniform distribution, low masses are slightly more common.

III. THE MODELS

Once we have simulated all the data, we can train our models with it. Before discussing the two models trained for this project, some context about DL, and CNNs in particular, is needed [14]. DL is a branch of ML based on the use of artificial neural networks, which are complex ML models formed by an input layer receiving information, some hidden layers performing different tensor operations and an output layer giving results. These layers can be connected in different ways, depending on the architecture of the network. They are called "neural networks" because they are inspired in how the brain works, although it is not clear at all that the brain works in the same way neural networks do. The basic unit building the layers is called neuron. Their first advantage in front of traditional ML models is the non-linearity of DL models, which makes possible modelling really complex systems. Probably, the most well-known kind of neural networks are CNNs. These networks, based on convolution operations (basically, matrix multiplications), are really good when working with images, and they have become almost ubiquitous in the present.

In order to train DL models, data has to travel the net back (back propagation) and forth (forward propagation) a certain number of times or epochs, which can be really time consuming. Nevertheless, the improvement of the GPUs has brought a golden era for neural networks, due to the time saving provided by parallelization of tensor operations. This has allowed both scientists and companies to create really advanced neural networks, such as VGG19 [15], a CNN trained with huge amounts of data. It would be impossible for an undergraduate student to train a CNN such as VGG19, but luckily we can count on Transfer Learning and Fine Tuning. These two ideas are crucial for this project. Basically, neural networks make their calculations based on a set of parameters called weights. During the training of a neural network, what we are doing is looking for the weights that minimize the error made by the network. Transfer Learning consists on downloading an already trained model, this is, a model with the optimal set of weights for a certain task. In the case of VGG19, this task is finding patterns in images. Once we have this pre-trained network, we Fine-Tune it by freezing part of the weights (i.e. they will not change during training) and training just part of the network with our own data, so that the network specializes in classifying it.

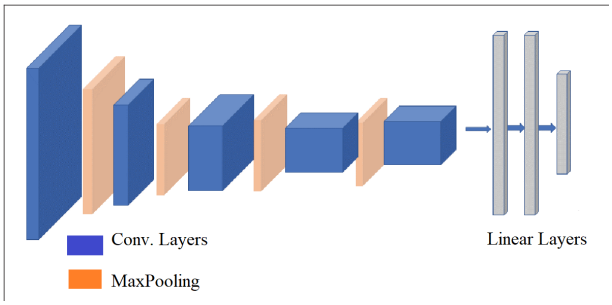

FIG. 4: VGG19 structure. It is made out of 16 convolutional layers, with some maxpool layers between them, plus 3 linear layers.

In our particullar case, the procedure has been the following: first, we download an already trained VGG19 network, with the structure of FIG. 4. Then we freeze the weights of the 16 convolutional layers, and change the 3 linear layers by 4 linear layers adapted to our problem (VGG19 classifies images in 1000 classes, but we want to create two models, that classify images in two classes each). Finally, we train the new 4 linear layers with our simulated data.

Following these steps, we create two models:

1. **S-N model:** Trained with around 18,000 noise images and 18,000 signal images (9,000 lensed and 9,000 unlensed) during 15 epochs. We use a learning rate of 1×10^{-5}.

2. **L-U model:** Trained with around 9,000 images of lensed singals and 9,000 of unlensed signals during 15 epochs. We use a learning rate of 1×10^{-5}.

In both cases, the number of epochs and the value of the learning rate (which is a hyperparameter that specifies the length of the step taken in each epoch when updating the weights) were chosen by trial and error.

IV. THE RESULTS

Both training processes took around 4 and 3 hours respectively. In the case of the L-U model, we got the loss and accuracy plots shown in FIG. 5:

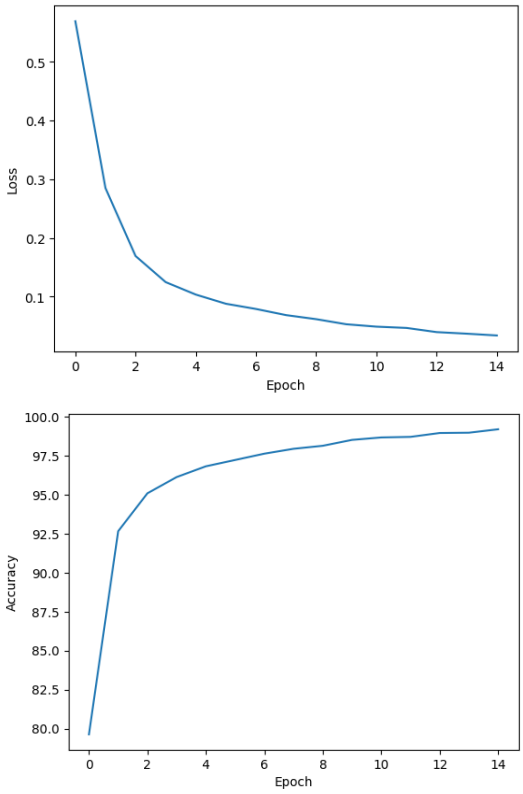

FIG. 5: **Upper figure:** Shows the evolution of the loss with the epochs for the L-U model. The loss measures the error of the network in each epoch. **Lower figure:** Shows the evolution of the accuracy with the epochs for the L-U model. The accuracy measures the percentage of correct classifications in each epoch.

Plots like these are what we expect when training neural networks: a smooth approach towards 0 in the case of the loss and to 100 in the case of the accuracy. The validation test helped us make sure that over-fitting was not happening in any of our models. When testing the models, we got accuracies of 100% for the N-S model and 98% for the L-U model, which are promising numbers. The reader can see the confusion matrices for both models in FIG. 6. We have to be careful, in any case, since our models may be experts in simulated data, but we do not know how they would perform with real data yet (this will be the scope of a future work). Some little modifications in the datasets may be needed. For example, we may need to apply a time shift in our signals, since all of them are set to be at $t = 0$. This would ensure that our model could detect shifted signals.

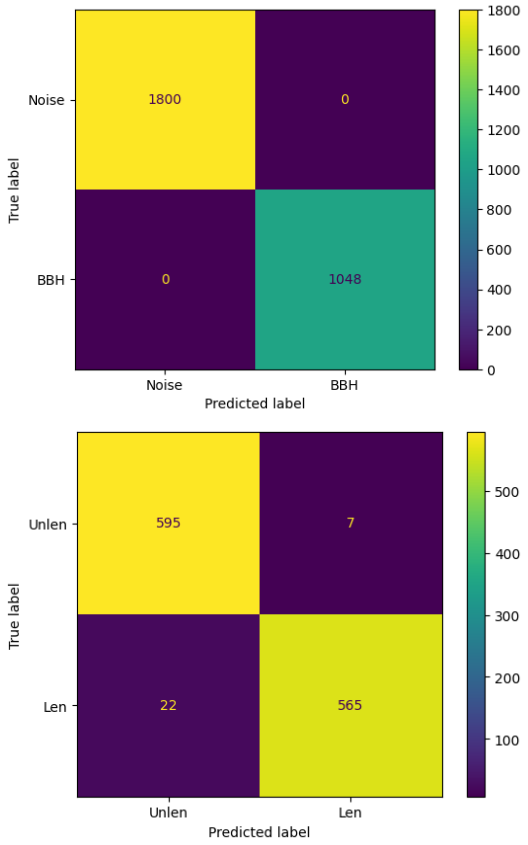

FIG. 6: **Upper figure:** Confusion matrix for the test set of N-S. We see that our model achieved a 100% accuracy. This is promising, since some of the images we used had a SNR of 5-6, and this signals could be hard to distinguish even for the human eye. **Lower figure:** Confusion matrix for the test set of L-U. We see that our model achieved a 98% accuracy, which is also a good performance.

We observe that N-S had a test set made out of 1,800 noise images and 1,048 signal images. In total, it analyzed 5,696 seconds of data in minutes, being able to identify every simulated signal, with no false alarm cases. This ratifies the useful ML models can be in the future when analyzing GW data, since they will save researchers an important amount of time. Similarly, in the case of the L-U model, the test set had 602 unlensed signal images and 587 lensed signals, and it was capable of classifying

them in minutes. In the case of the lensed class, the model only had a 1.2% false alarm ratio.

V. THE CONCLUSIONS

In the light of these results, we state the following conclusions:

1. ML methods are capable of carrying classification tasks in the context of GW search in a fast and accurate way.

2. In the particular case of the study of time-frequency spectrograms, we find that CNNs such as VGG19 perform specially well when differentiating between noise and simulated signals with injected noise.

3. Also in this context, VGG19 has proved to distinguish quite well (98% accuracy) between simulated lensed and unlensed signals. This is specially exciting, since lensed GWs are yet to be discovered.

4. One of the main strengths about this type of models is how fast they prove to be in detection tasks. This characteristic of these models make them perfect for being combined with traditional GW search methods such as MF, which can be excruciatingly time-consuming.

Final note: The models were trained by the GPU NVIDIA V100 TENSOR CORE provided by Google Colab.

Acknowledgments

Thanks to my parents for maintaining me, to Laura for her patience and to my grandmother for feeding me during the hardest days of work. Special thanks to my advisor, Professor Oleg Bulashenko, for always being available and kind; and to my internship advisors Alejandro and Montse, for always having time to solve my doubts in AI. Also thanks to Tyler, JP and Dani for their useful comments.

[1] R. Abbott et al. (LIGO Scientific Collaboration, Virgo Collaboration and KAGRA Collaboration), "Open data from the third observing run of LIGO, Virgo, KAGRA and GEO", arXiv:2302.03676 (2023).

[2] R. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), "Open data from the first and second observing runs of Advanced LIGO and Advanced Virgo", SoftwareX **13**, 100658, (2021).

[3] B. S. Sathyaprakash and S. V. Dhurandhar, "Choice of filters for the detection of gravitational waves from coalescing binaries." Phys. Rev. D **44**, 3819 (1991).

[4] Kim, K.; Harry, I.W.; Hodge, K.A., et al. "Application of artificial neural network to search for gravitational-wave signals associated with short gamma-ray bursts." Class. Quantum Grav. **32**, 245002 (2015).

[5] Jiang, MQ., Yang, N., Li, J. "Identify real gravitational wave events in the LIGO-Virgo catalog GWTC-1 and GWTC-2 with convolutional neural network." Front. Phys. **17**, 54501 (2022).

[6] K. Kim, J. Lee, R. S. H. Yuen, O. A. Hannuksela, T. G. F. Li, "Identification of Lensed Gravitational Waves with Deep Learning." Astrophys. J. **915**, 119 (2021).

[7] K. Kim, J. Lee, O. A. Hannuksela, T. G. F. Li, "Deep Learning-based Search for Microlensing Signature from Binary Black Hole Events in GWTC-1 and -2." Astrophys. J. **938**, 157 (2022).

[8] Shang-Jie Jin, Yu-Xin Wang, Tian-Yang Sun, Jing-Fei Zhang, Xin Zhang. "Rapid identification of time-frequency domain gravitational wave signals from binary black holes using deep learning." arXiv:2305.19003 (2023)

[9] M. Zevin, S. Coughlin, S. Bahaadini, et al. "Gravity Spy: Integrating Advanced LIGO Detector Characterization, Machine Learning, and Citizen Science." Class. Quantum Grav. **34**, 064003 (2017).

[10] S. Chatterji, L. Blackburn, G. Martin and E. Katsavounidis, "Multiresolution techniques for the detection of gravitational-wave bursts", Class. Quantum Grav. **21**, S1809 (2004).

[11] J.C. Brown, "Calculation of a constant Q spectral transform." J. Acoust. Soc. Am. **89**, 425 (1991).

[12] A. Nitz, I. Harry, D. Brown, et al. (2023). gwastro/pycbc: v2.1.2 release of PyCBC (v2.1.2). Zenodo. `https://doi.org/10.5281/zenodo.7885796`

[13] R. Takahashi and T. Nakamura, "Wave effects in the gravitational lensing of gravitational waves from chirping binaries", Astrophys. J. **595**, 1039 (2003).

[14] A. Ng, T. Ma, "CS229 Notes" (2023), `https://cs229.stanford.edu/main_notes.pdf`.

[15] K. Simonyan, A. Zisserman "Very Deep Convolutional Networks for Large-Scale Image Recognition " International Conference on Learning Representations. arXiv:1409.1556 (2015).