CARRERA: ESPECIALIZACIÓN EN CIENCIA DE DATOS

TRABAJO FINAL INTEGRADOR

"Comparativa de rendimiento de clasificacion de espectrogramas de ondas gravitacionales mediante la utilizacion de tecnicas de machine learning"

Nombre y Apellido del Alumno/a: Msc. Ezequiel H. Martinez

Título de grado o posgrado (último): MBA

Tutor:

Dr. Rodrigo Ramele

Lugar y Fecha: Buenos Aires, 06 de Septiembre de 2020

CAREER: DATA SCIENCE SPECIALIZATION

FINAL WORK

“Analysis and benchmarking for gravitational waves spectrogram’s classification by usage of machine learning techniques"

Student: Msc. Ezequiel H. Martinez

Maximum Degree: Master of Business Administration

Tutor:

PhD. Rodrigo Ramele

Date & Place: Buenos Aires, September 6th, 2020

Table of contents

1. Abstract

Gravitational waves, the seed of the 2015 Nobel's prize are the cause of several complex celestial phenomena that is non-observable for the naked eye. Their identification, classification and study is still a handmade work which is still nascent. There has been several approaches to produce novel tools to aid the scientists behind the discovery of these deep space events. One of the most thrilling examples has been the usage of artificial intelligence classification to aid in the pre-identification of certain signals. We took one of these tools, Gravity Spy, and study its base paper, trying to reproduce some of their classification results using the very same base dataset.

This research aims to compare the results obtained from the original paper, with a binary-classification approach and several different algorithms taken from the knowledge base of machine learning and deep learning, alike. We confirmed the original paper results and obtained a new approach for the same solution. In this study we trained several models that could be used for further development of an eventual alternative engines for gravitational waves signal's classification or any other sort of signal heavily influenced by noise and analysed by spectrograms.

2. Introduction

The Advanced Laser Interferometer Gravitational-Wave Observatory (LIGO) has opened the field of gravitational wave astronomy through the direct detection of signals predicted by Einstein's General Theory of Relativity. Advanced LIGO's first observing run (O1) saw the first detections of binary black hole mergers. Advanced LIGO and Virgo's second observing run (O2) included both binary black hole and binary neutron star mergers.

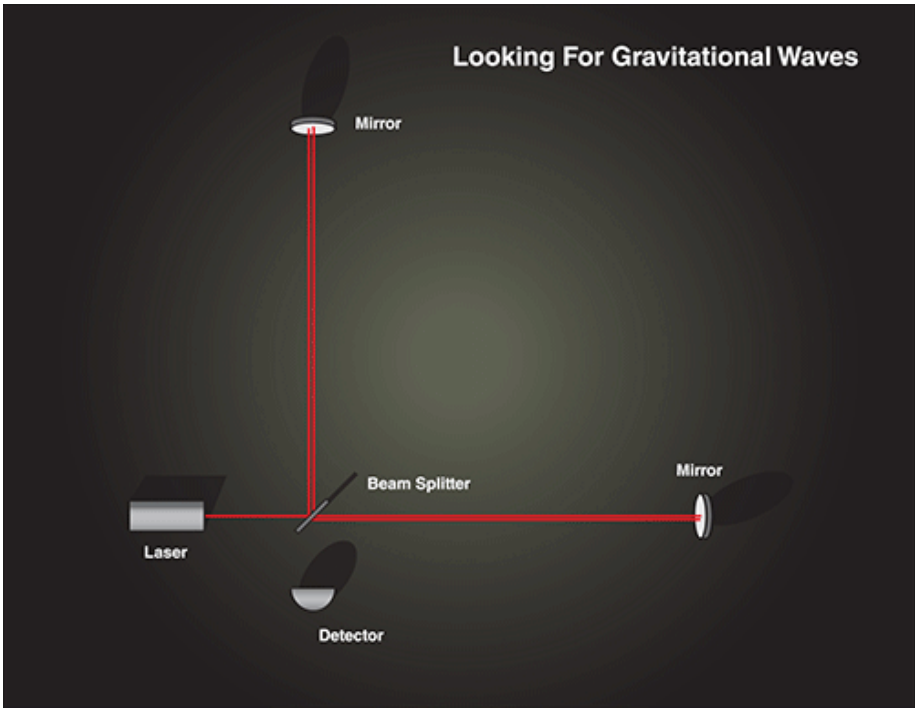
Gravitational waves are signals emitted by objects of high mass that resides in deep space. Usually, they can be emitted in the occurrence of a celestial event. They constitute disturbances in the space-time fabric. These object are typically massive and of the size of a star. These signals occurs in a moment at which there is heavy activity, which could be:

- A Supernova explosion
- A merging of two binary stars.
- A merge of two binary black holes.
- Other types

These signals are measures by a specific kind of massive observatory named LIGO[1]. The first successful readings were taken during 2015[2]. These measurements are organised in blocks of data that are being structured in different datasets. Some of these datasets are freely available and are related to a specific time and place. Despite the fact that not all the data is openly available; humanity count today with around four years of measurements[3] organised in several rounds of observations.

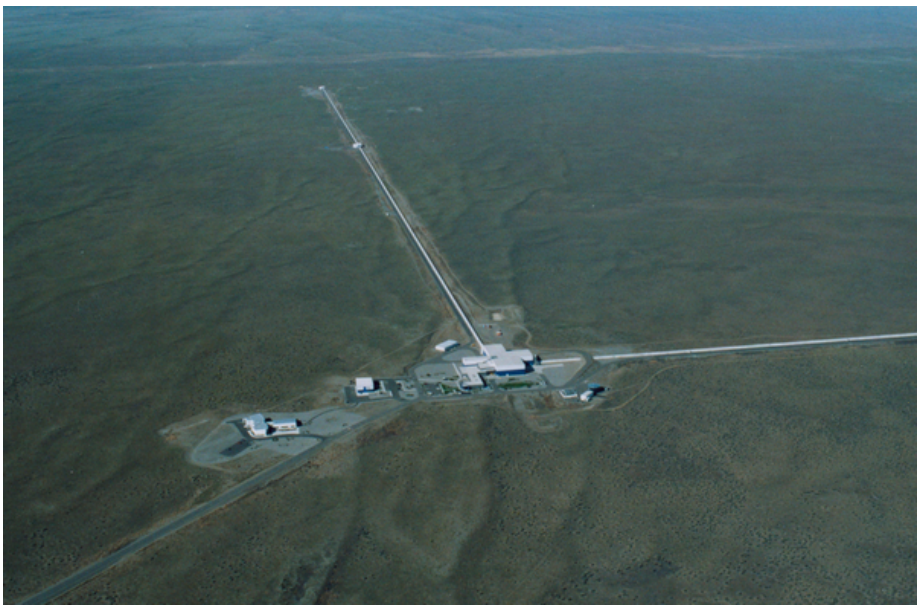
The instruments at the core of these readings that take those signals from deep space are called interferometers[4] and work inside the LIGO observatories. An "interferometer" it is analogous to a telescope for electromagnetic signals. As the name states, is a sensor designed to identify interferences. There are of several sizes, but the ones you can find in Livignston and Handford observatories are particularly big. Size matters, and their intent is to catch the specific interferences produced by a gravitational's events of magnitude.

Figure 2.1 - The general internal organisation of a LIGO interferometer. [5]



As we can see in figure 2.1, they are built with a single laser beam that is being divided between two rays. The variations in length and frequency of any of those two arms in respects to the other produces the detection of a possible valid signal.

Figure 2.2 - LIGO is made up of two observatories: one in Louisiana and one in Washington. Each observatory has two long “arms” that are each more than 2 miles.[6]



Since an interferometer catches interferences, they are highly sensitive to all sort of white noise. There are techniques and challenges associated with the preparation of these measurements. One of the few being the signal's filtering and cleanse[7]. After proper treatment, the signals taken by the sensors are digitalised for its later analysis. They are structured in "data-burst" that are measured by fractions of a second[8].

In spite that some of the historical data is available and refers to past observation's runs, the complete current dataset isn't open to the public. These are data already partially classified and treated. In this sense, we will be able to work with data that is already classified and tagged by scientists.

Currently, the data is generated from four different observatories:

- `'G1'` - GEO600
- `'H1'` - LIGO-Hanford
- `'L1'` - LIGO-Livingston
- `'V1'` - (Advanced) Virgo

These sensors create datasets from the same events, measured by different latitudes and longitudes from earth. These varied measurements help to apply techniques that could help in the identification of miss readings or false positives in the identification of waves. There are some specific signals that looks very closely to what a gravitational wave might look like, they are referred as glitches.

The glitches can be defined as peculiarities in the signal that simulate o are similar to what a gravitational wave might look like. [9] These are considered "false positives" and supposes one of the most dreaded occurrences to deal with during the study of these events. Still, they are fairly common.

All these measurements has been standardised and resolved in structured datasets that can be consumed by a regular Python script. There are in place several efforts to work in the readings and cleaning of the signal out of noise[10]. We understand that there are a variety of methods to test and apply for this to be accomplished; we know that some of the techniques utilised make use of a technique denominated Deep Filtering[11] which is making usage of convolutional neural networks and other machine learning algorithms to do so. When not, we used some of these papers as inspiration for the present work.

In spite all this, data around these discoveries are partially publicly available, along with associated software libraries. This work is based in one dataset and its paper: "Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science"[12], published over the classification of "glitches" that affects those waves' readings and the project developed around it: Gravity Spy. All of which was designed for scientists and students pursuing research in this field, both inside and outside the LIGO Scientific Collaboration. This paper speaks of a dataset of pre-classified spectrograms over these waves. They were classified by a public science effort made possible by crowd classification[13]. This dataset and paper provides us with info over:

- A particular kind of "glitch": the "chirp", which was identified as being to most close to a gravitational wave form.
- Assesses the performance of different "Machines Learning" techniques in order to classify these spectrograms as specific glitches occurred during readings.

We will then train several models that will classify just one of the glitches: the chirp. Given this, we'll be trying to approximate the results that the author of the original paper achieved. Besides this, our main intent will be to assess these algorithms performance in comparison with each other. Taking relevant metrics and assessing the results.

A few questions that we want to answer are:

- How hard might be for some of the presented algorithms to identify a gravitational wave given its due spectrogram?
- Can we approximate the classification of the same dataset, when it comes to "chirps" to what the Gravity Spy has achieved?
- Can we validate alternative ways to classify "chirps" that weren't present in the original paper?
- Can we be able to train a model that classifies at least one "Chirp"?
- It is the same to classify the gravity spy's dataset with a deep learning technique than with a more traditional machine learning technique?
- These spectrograms supposes a un-structured dataset, given that they are images, or its data is truly regular?
- Can we arrive to some of the same conclusions or validate some of the conclusions obtained by the gravity spy paper?

Some of the contributions of this work are expected to be:

- Provide a comparison of several models' performance to bring more light over the classification and characterisation for noisy signals when represented as spectrograms.
- It will provide several models capable of classifying a spectrogram in a binary way: either a candidate gravitational wave or not.
- Provide a cue about how to automatically identify candidate gravitational waves throughout the usage of spectrograms.
- Give another glance of how to deal with these signals, providing an already trained model to try for binary classification of newer signals.
- Deliver a trained model that could help scientists to classify specific signals, against all the others.

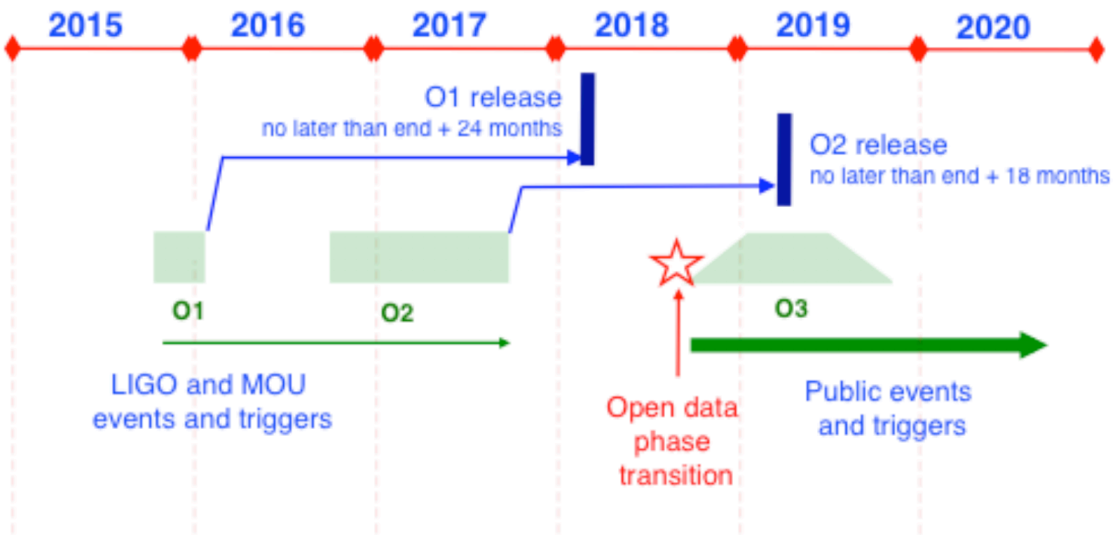
- Verification through reproduction of the findings found in the Gravity Spy paper is some of the due diligence expected in science.

3. Challenges

There was a challenge held for the LIGO open data workshop of 2020[14], in which with a limited and open held data, the attendees tried to classify several binary black holes generated by the scientists that held the workshop. The aim was to allow students and senior scientists to learn about the datasets structure and software tools[15] involved in the work that the main scientist involved in the research for LIGO are currently doing. The aim in those workshops was to train enthusiasts and physicists in the classification methodology commonly used to identify valuable signals. The difficulty to classify these waves isn't trivial, given the dataset's size[16] and the technical complexity needed to filter and evaluate the samples up to a point at which it can be understood[17]. Therefore, developing a technique that could aid in the classification of candidate gravitational waves could be of great value for the average astrophysicist.

One of the issues for an automatic method to succeed has been the availability of a pre-classified datasets with which to train a model. The amount of recognised celestial events is scarce, and the training is done with artificial built up signals; as we can see from the paper ""Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data"[18]. More over: the complete most recent readings aren't publicly available for the whole scientific community as they are being produced. There is a plan in place to release these measurements that usually felt several months or years after they were taken[19].

Figure 3.1 - The current data release plan for LIGO data measurements[20].



We got a dataset of pre-classified spectrograms that we might use as a base study to show and study how the model's used in a project like Gravity Spy[21] could have been developed. Since the dataset for the same paper is publicly available; to reproduce some of these findings between different techniques could end providing another tool to use for the people involved in the classification of these signals. There has been different efforts for classification of these waves, like deep filtering for sound signals[22] or deep learning for real-time gravitational waves[23], still most of the work is being done manually and with statistical methodologies.

In this work, we are going to train four models with a pre-classified dataset of real readings which were manually classified as glitches by a crowd of scientists. This is the dataset provided by the "Gravity Spy paper". We will use it for the intent of evaluating four algorithm's performance in the classification of the "chirps" sawn in the same paper. We are going to aim to a "binary classification" of the kind: "chirp / gravitational wave" vs the rest. Since our approach is to analyse a one-vs-all kind of classification, it could open the door for comparison against the process performed by Gravity Spy. That paper classified every category in a non-binary non-iterative way. The main difference between this approach and the general classification made by Gravity Spy is that we might find suboptimal algorithms that could miss classify some signals for the "chirp" category. Having some signals "approximately" classified as "chirps" we could bring insights about some other sort of signals, not yet discovered, that could lead a trained human observer to a breakthrough discovery.

Moreover, given this idea in place, a future researcher could try to go for binary-classification of all the glitches in the dataset. Evaluating the spectrograms nature, similarities and characterisation in a different view.

However, these are side goals of this work. Out main goal is to evaluate these four algorithms and compare their performance against the findings in the "Gravity Spy" paper.

4. Methodology

The main methodology used for this work involves different machine learning and deep learning algorithms comparatives. We are going to use the following algorithms and measure its performance trying to execute a binary classification of spectrograms of half seconds for signals taken from deep space.

4.1 Methods and Materials

Algorithms and model's sources

For this research we will be able to use the scikitlearn[24], pandas[25], keras[26] and numpy[27] libraries provided by the body of tools found in the Python's universe. We've worked with four different algorithms:

Machine Learning algorithms[28]

- Linear SVC: the core of a support vector machine, specialised in binary classifications. It uses the concept of an hyperplane that "divides" the dataset in two using several "support vectors" distances to it as minimum factor for classification of every observation. These are good image classifiers.
- Light GBM: (Gradient Boosting Machines) gradient boosted classification is a "voting" kind of training tree classification where trees are built in series and compared to each other based on their scores. The winning classification is evaluated on weighted leaf scores within each of the best performing trees.

Deep Learning algorithms[29]

- Convolutional Neural Networks: a neural network that has other neural networks in its inner layers, feeding the following layers with the output of each one. They contain many convolutional layers over each other, each one capable of classifying more complex data. These networks are said to be good to recognise images and other complex patterns.
- Recurrent Neural Networks: Convolutional Neural Networks that introduces the concept of "memory" in their design. They refine every classification based on their own performance in the "past". These are good for classification of time series data.

We have developed several models to work with everyone of them. Since the dataset has been constrained to just spectrograms, we are classifying these images regardless of their time series natural scope. The base for these model's training and validation of the datasets are the methods suggested by François Chollet[30] and James Gareth "et al"[31] which entails the training of different deep learning algorithms and machine learning ones by dividing the dataset in three layers:

- A **training** set: where we will be taming the models for binary classification.
- A **validation** set: where the trained models will be validated during training, if applicable.

- A **testing** set: where we will evaluate the model's performance and therefore raise metrics.

(We are going to use the division sets used in the "Gravity Spy"[32] paper, since the same comes already divided and tagged in these three subsets. Besides, we wanted to keep the experiments as similar to the original paper as possible.)

After this step was taken, we configured the different algorithms, trained them, saved the model and measured its performance.

Algorithms metrics

The general metrics we have in mind to evaluate performance are the following:

- To verify that the trained model classifies a "Chirp" class (a Gravitational Wave).
- Usage of "Loss" and "Accuracy" functions in order to estimate how "good" the trained model is.
- ROC's area under the curve (AUC).
- Precision.
- Recall.
- Times to train.

We haven't used any specific of technique for hyper parameters search, aside from the ones proposed in the mentioned bibliography. For more info about the metrics used, please refer to **Appendix B.**

Research's experimental code base

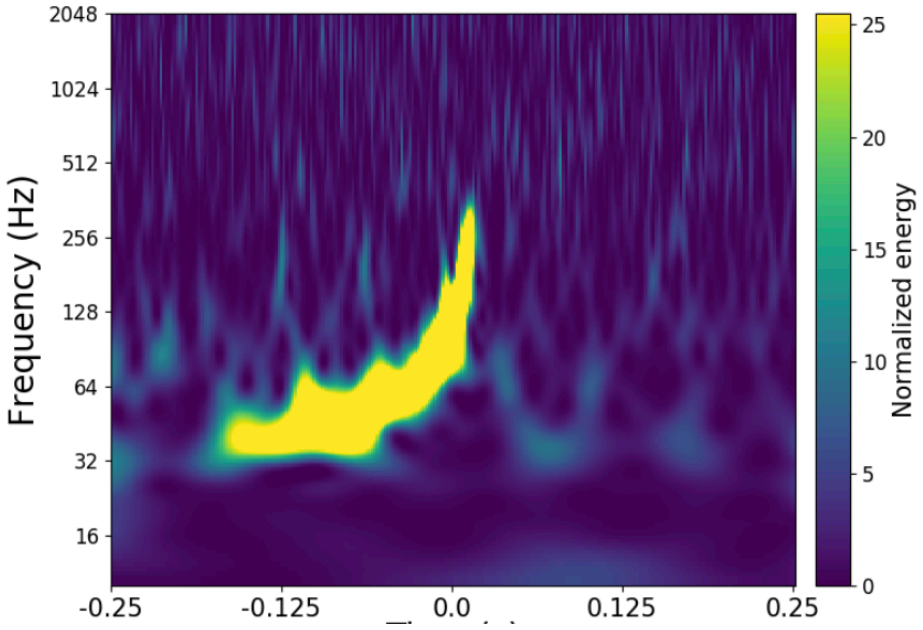
All the algorithms and experiments performed during this research could be found in: https://github.com/exemartinez/gravitational_waves_classifiers/tree/master/gw

4.1.1 Introduction to the GW spectrograms

The images we will be classifying are spectrograms; this means they are signals representation in a constrained span of time by its intensity. In order to properly understand the results we will be taking, we need to understand how they look and how the transformation work over them went.

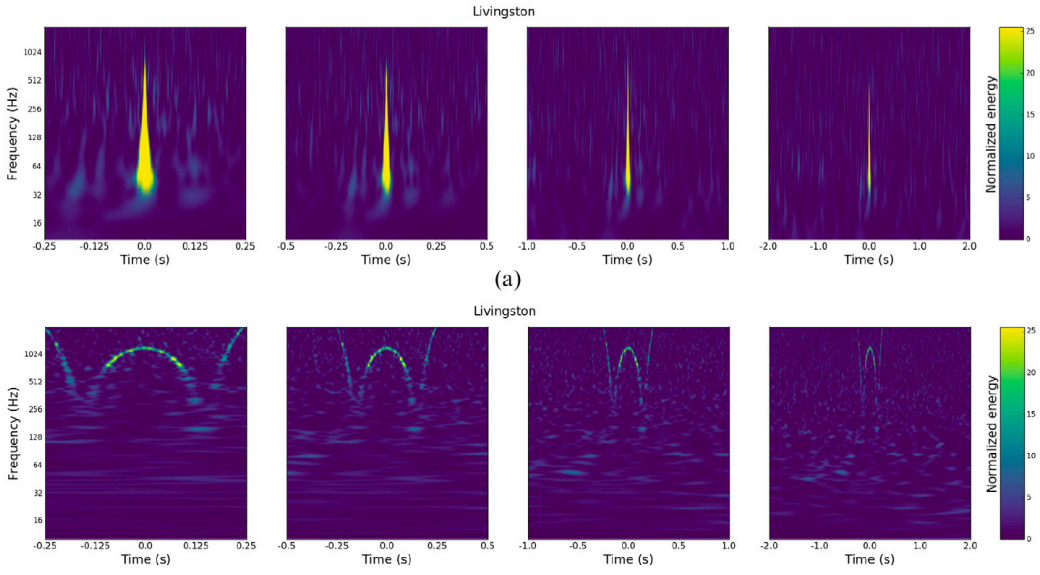
A gravitational wave looks like a "chirp" in the spectrogram, such a signal looks like the one represented in figure 4.1.1.1, which was recovered from the dataset we'll be dealing with.

Figure 4.1.1.1 - This is a "Chirp", a signal candidate to be classified as a gravitational wave by a Physicist.[33]



There signals will be compared and classified with other images of the same kind, like the ones described in the Gravity Spy paper:

Figure 4.1.1.2 - Several spectrograms examples that aren't chirps neither gravitational waves.[34]



In figure 4.1.1.2 we see the different kind of images and how their timeframes affect their look; we'll deal only with images in the 0.5 seconds span and analyse the algorithm's findings.

In spite of the appearance of the spectrograms, we need to remember that we will be classifying the 140x170 matrix that defines those images. We checked these and discovered that there are only data related to every pixel of the main images; as we discussed in **appendix A**.

Something else to take into account from the base dataset is the analysis and results obtained during the "Gravity Spy" research. We will use these as a benchmark to which we could compare our results, more specifically, the "Chirp" row/column in comparison with everything else. These results could be found in figure 4.1.1.3, where the author sustains the confusion matrix[35] for the CNN classification he made over the same dataset; we'll be doing the same thing within a binary approach.

Figure 4.1.1.3 - The confusion matrix for the classification of the 20 glitches with a trained CNN[36].

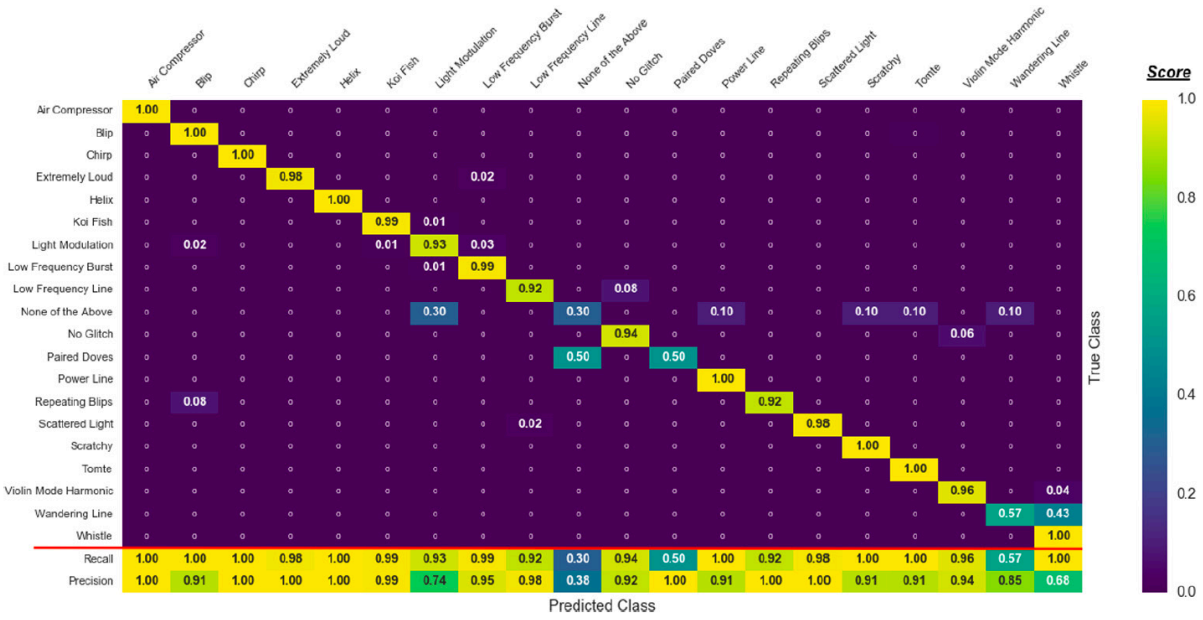


Figure 4.1.1.3 shows the precision and recall for every given class (see **Appendix B** for more details on the metrics). Here, we count with 18 different classes with a multi class analysis. In a confusion matrix, the main diagonal represents "how well" that particular class was classified (ranging from 0 to 1, higher the better). Numbers out of the diagonal represents elements that were miss classified. This image is important because we base all our comparison analysis over these result**s**. As a matter of fact, the metrics we use in this research are either these same metrics or a a derivation of them:

- The confusion matrix is the base for he ROC/AUC.
- The precision and recall, composes the f1 metric.

- We added some other metric for “inter model” comparison when applicable, like accuracy and loss.

Keep in mind that we will base all our analysis over the performance of the category “chirp” alone.

The main take away here is this: the algorithms used in the paper achieved perfect score for the “chirp” category. This is remarkable and we tried to approximate these results in our work.

4.2 Classification through Linear SVC

Hardware and configurations pre-sets

This work has been performed over a home hosted virtual machine with the following configuration:

- 2 Cores
- 16 GB of RAM
- Linux Ubuntu 19.0 (workstation) as Operating System

Algorithm's models and evaluated results

The Linear SVC (Linear Support Vector Classifier) it's a kind of support vector machine designed to deal with binary classification of complex data. It is a special kind of support vector machine with a linear kernel and a distance from the hyperplane to classify two classes[37].

We used the Sci Kit Learn library[38] to test this classification and draw metrics from it. The configuration we build for our first classification upon it was as follows:

- A linear kernel.
- A "C" parameter 1.0 (the span distance to the division hyperplane supported by the vectors).
- We trained the algorithm within the original "train" label, composed of 5,587 float matrices that corresponds to every spectrogram. We knew that in this set we'll face roughly 41 "Chirps".
- We provided a balanced "class_weight" param; since we know that the chirps supposes less than the 1% of the observations in the set.

We trained this model and then validated it. We used the standard way of training suggested by the library and then we used cross-validation[39] over five epochs. The validation set was the very same one labeled and suggested "as is" in the "Gravity Spy" paper; this was intentional due to the fact that it will allow us to contrast findings and classification results over the former ones.

Table 4.2.1 - The 1st model prediction against the "validation" set

Label	Precision	Recall	f1	#
0	1	1	1	1191
1	1	1	1	9
ROC/AUC	1			

Alongside with what we can see in table 4.2.1, the ROC/AUC metric was of 1 as well. It probed to be a "perfect score" that was checked and confirmed by its performance over the test set suggested by "Gravity Spy". It classified all the spectrograms without a mistake. We performed several other configurations over the same training set in order to try to understand this behaviour, since an algorithm that classifies with a AUC/ROC score of 1.0 is suspicious of errors or even overfitting. However, as can be observed in Figure 4.1.1.3 (the confusion matrix obtained in the gravity spy classification by a CNN.), we see that the classification performance of "chirps" is 1.0. Which is encouraging and gave us a cue about the current set's nature. These findings could be reproduced executing the scripts found in the code base[40].

Table 4.2.2 - The model classification of the validation and test set got the right images.

	No-GW	GW	Identified
Validation Set	1192	9	9
Test Set	1169	10	10

As we can see in table 4.2.2, all the images were correctly recognised.

To understand this performance and the rationale behind it, we performed several other trainings and classifications. For our 2nd model we changed the following:

- We took all the 41 "chirps" from the gravity spy "train" set and mixed them together with 41 spectrograms randomly extracted from the same set (taking care in not picking another "chirp", also).
- We reduced the C parameter progressively from 1, leading it all the way down to 0.000075.
- We validated the set without cross-validation and then check it against the labeled "test" set.

The results we found where the same as long as the C parameter was over 0.000075; as far as we approximated this hyperparameters to this value or surpassed it the entire model precision, recall, ROC/AUC and f1 parameter started to deplete.

Table 4.2.3 - Once the C parameter approximated to 0.000075 the metrics fell.

Label	Precision	Recall	f1	#
0	1	0.94	0.97	1191
1	0.12	1	0.21	9

Table 4.2.3 shows how the metrics started to get down. Therefore, the model prediction performance in the original "validation" and "test" set were as the ones showed in figure 4.2.4.

Table 4.2.4 - The prediction's performance turned out to "predict" much more false-positives after we diminished the C parameter.

	No-GW	GW	Identified
Validation Set	1192	9	78
Test Set	1169	10	67

As we sae in table 4.2.3 the f1, precision and recall, all still sustained a pretty high score. The ROC / AUC dropped only to 0.88. However, the false positives identified by the model, as we see in table 4.2.4 soared. (All these metrics and what they do represent has been described in **Appendix B**).

We tested several other approaches in order to understand why this behaviour happened, as such:

- We mixed the labels and observed the metrics, under the guess that something could be oddly configured or that the library's version could be wrong or "bugged" (trying to implement an informatics equivalent to a measurement's error avoidance technique[41]). The metrics ROC/AUC, Precision, Recall and F1 were below 0.35 approximately.
- We tried to use our very own training and test sets, and ended up with similar results across all the training experiments previously tested.
- We tried with different values of C: 0.5, 0.1. These maintained the ROC/AUC, precision, recall and f1 performance that we found when we used 1 as the parameter. With values like 0.0005 and 0.000075 or below the model started to miss classify again.

The training code for the final algorithm, can be retrieved from the code base[42]. The predict's scripts and their metrics can be found from the code base[43].

Table 4.2.5 - The times for the optimal configuration

Linear SVC	Training Times	Predict Times
Train Set	76 secs.	NA
Validation Set	3 secs.	3 secs.
Cross Validation	60 secs.	NA
Test Set	NA	3 secs.

We conclude that the spectrograms constitute a very regular set of samples. These are bicolor, centred and regular images. Given those, and considering the small magnitudes at which the hyperplane drawn by the algorithm was starting to falter (below 0.0005); it gives us the idea that

the support vectors between the chirps and every other image are very far away from one another. Which rides us to the conclusion that the dataset, for the given classes and the available data, has more structure than other typical classified images' dataset (like the ones found in the MNIST dataset[44], for example, which has been broadly studied[45]).

It is worth to notice the regularity of metrics like the precision, the recall or the F1, which are interrelated and represents the model's "accuracy". Those metrics, alongside with the ROC/AUC, never go below 0.8; which is a fairly high score. The reason for this to happen, we conjecture, has to be with the fact that the cardinality of the class-1 ("Chirps") images are far below that of the class-0. As a result, these metrics became altered in the sense that the "non-gw / class-0" is massive and easily identified.

4.3 Convolutional Neural Network Approach

Hardware and configurations pre-sets

This work has been performed over a home hosted virtual machine with the following configuration:

- 2 Cores
- 16 GB of RAM
- Linux Ubuntu 19.0 (workstation) as Operating System

Algorithm's models and evaluated results

We applied here the given sub-set of training pre-classified samples; they were 5587 images. We configured the network with different layers; which were inspired on an exercise from the Deep Learning book[46] example for classification of digits of the MNIST classic dataset:

Layer Type	Outputs	Activation
Conv2D	32	Relu
Conv2D	64	Relu
Conv2D	64	Relu
Dense	64	Relu

The "Conv2D" is a convolutional neural network organised for a 2D matrix. The amount of outputs are the values that will go for the next layer. The "Dense" layer, represents a regular neural network layer. The activation is the kind of function that transforms the score output in every exit as a classifier.

We configured the optimisers and loss functions as "RMS" and "Binary Cross Entropy" and trained over the same dataset over five epochs.

This network failed to classify the images; in spite the fact that the "accuracy" metric stayed over 0.992 with a "loss" below 0.007. The small amount of error remained still higher than the percentage of "gravitational waves"(class-1) spectrograms; which were below the 0.008 of the total. We then trained over 50 epochs within the same results. We used these metrics just as a first approach to the set. In spite the high value metrics the model failed to identify a single gravitational wave in any of the "validation" or "test" sets.

Later, we found that the network "deepness" was the problem. With the given positives' set (41 in 5587), we needed to build up a bigger network. Therefore, we changed the model for a similar one found in the same book[47] for classification of a small set of images of "dogs" vs "cats" (The dataset

was small, roughly 2000 images, but the categories remained binary as well). We adapted such a model in the following way:

Layer Type	Output	Activation
Conv2D	32	Relu
Conv2D	64	Relu
Conv2D	128	Relu
Conv2D	128	Relu
Dense	512	Relu
Dense	1	Sigmoid

We added three more layers; and changed the activation for the last layer to “sigmoid”. This network showed the following performance during training:

Table 4.3.1 - Generated during the training session for the Convolution Neural Network approach.

Accuracy	0.000121
Loss	0.99915

After we trained the network for 30 epochs, the trainer algorithm found that the loss and accuracy ceased to improve after the ninth epoch. Therefore, we tested the model against the “testing set”, which thrown the results shown in table 4.3.1.

Figure 4.3.2 - ConvNet accuracy vs validation set

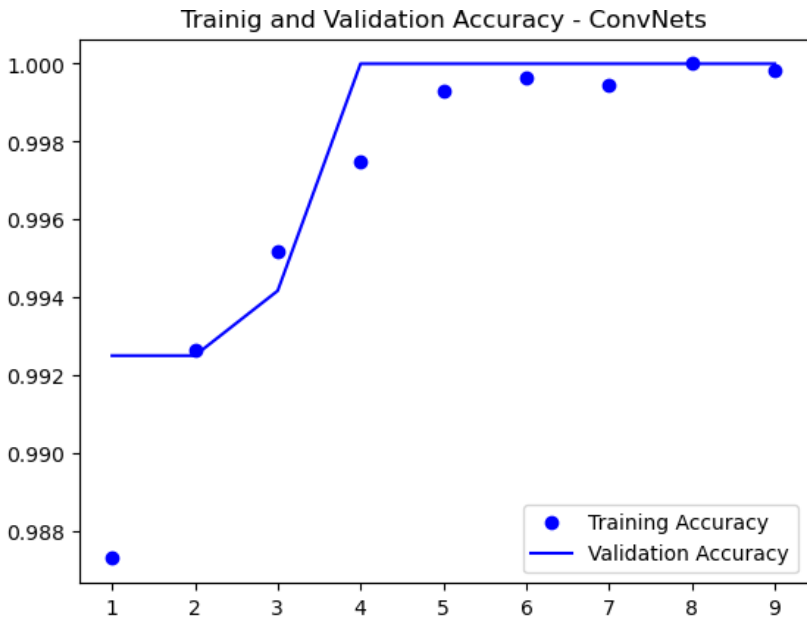
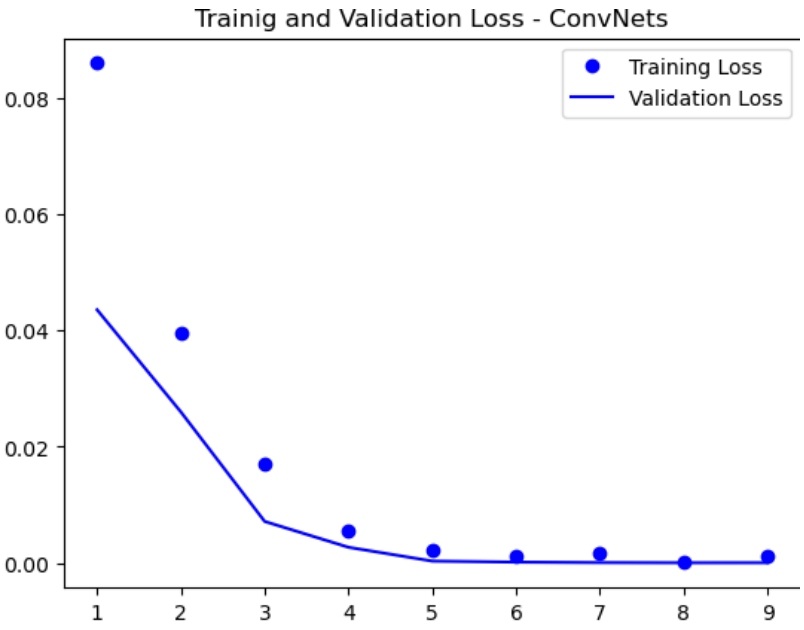


Figure 4.3.3 - ConvNet loss vs validation set



The history for these epochs, as can be appreciated in both images (Figure 4.3.2 and 4.3.3), shows that the training and validation sets converges during cross-validation training for almost all nine epochs; which ensures that we are avoiding overfitting issues.

Table 4.3.4 - ConvNet precision, recall, f1 and ROC/AUC for the "validation" set.

Label	Precision	Recall	f1	#
0	1	1	1	1191
1	1	1	1	9
ROC/AUC	1			

Table 4.3.5 - ConvNet precision, recall, f1 and ROC/AUC for the "test" set.

Label	Precision	Recall	f1	#
0	1	1	1	1169
1	1	0.90	0.95	10
ROC/AUC	1			

As can be sawn in Table 4.3.4 and 4.3.5, the second convolutional neural network's model reached an asymptotic curve around epoch 5; we saw afterwards that the network classified all the validation set's gravitational waves (9 out of 9) and most of the same images from the tests set (9 out of 10). The ROC/AUC curve was 1.0 for both sets.

Table 4.3.6 - The times for the optimal configuration

Convolution Neural Network	Training Times	Predict Times
Train Set	20 minutes	NA
Test Set	NA	26 secs.

The performance for the trained algorithms seems to reach a "perfect" score of one in almost every metric. As such, we understand that the peculiarities of the dataset might be the reason for this behaviour: the signals are very regular and with little to no change. The fact that the ROC score is one means that there are no "false-positives". Since it is an interest of any scientific endeavour to identify new events, having a perfect score could be a symptom of "overfitting" in a more general/ business-like approach.

However, given the regularity of the set and the scarce class-1/gravitational waves in it, we see that not "any" model could do the match. With the first model, the metrics for accuracy, loss, were fairly high, but still not a single image was correctly classified. We understand this effect was due to the fact that class-0 was vast and easily identifiable. For the second model, we changed the network's deepness and evaluated the precision, recall, f1 and ROC/AUC. All of them, different views of the same measure: how well the true positives are being identified in the trained model.

4.4 Recurrent Neural Network Approach

Hardware and configurations pre-sets

This work has been performed over a home hosted virtual machine with the following configuration:

- 2 Cores
- 16 GB of RAM
- Linux Ubuntu 19.0 (workstation) as Operating System

Algorithm's models and evaluated results

We decided to test a LSTM model. However, a RNN works best when classifying time series kind of data instead of images; since a spectrogram is based in a kind of progressive data, we adapted the RNN to take the abscise axis as the time dimension and trained the models under this assumption. With this approach we classified the "Chirps" (gravitational waves) as in the other methods. The configuration will entail the usage of a LSTM (Long-Short Term Memory) configuration[48]. We used the following configuration in a "sequential" model:

Layer Type	Outputs	Activation
LSTM	4	NA
LSTM	4	NA
LSTM	4	NA
Dense	1	Sigmoid

The "LSTM" is a recurrent neural network organised as a long-short term memory. The amount of outputs are the values that will go for the next layer. The "Dense" layer, represents a regular neural network layer. The activation is the kind of function that transforms the score output in every exit as a classifier.

We used three LSTM RNN's layers in sequence with an sigmoid activation function. The model was compiled with the following parameters:

- optimizer = optimizers.RMSprop(lr=1e-4) - A Root Mean Square optimizer[49].
- loss = 'binary_crossentropy' - Using the loss as Binary Cross Entropy[50].
- metrics=['binary_accuracy']

And trained with the following parameters:

- epochs = 90 - we instructed the algorithm to fit during 90 epochs
- patience = 3 - If one epoch doesn't improves three times in a row, we cease the training.

- batch_size=100
- validation_data= used the one proposed by Gravity Spy.
- class_weight = we balanced the classes weights with the balance we identified from the dataset (0.008 class 1, and 0.992 for class 0)

In the 69th epoch, we reached the peaks of improvement for the model. With the results showed in table 4.4.1, accuracy above 0.99 and loss below 0.11; the Recurrent Network returned prediction's probabilities for each of the given binary classes. After try-and-test, we decided to assign a threshold of 0.75 for the class 1 (the "Chirp") and take a binary classification over it.

Table 4.4.1 - The Recurrent Network, first classification model loss and accuracy.

Loss	0.1025
Accuracy	0.9957

In spite the fact that those metrics seems to be very high, they are non-conclusive. These were later checked on the "predict" phase.

Tables 4.4.2 - The trained LSTM RecNet performance results.

Validation Set

Label	Precision	Recall	f1	#
0	1	1	1	1191
1	0.9	1	0.95	9
ROC/AUC	0.9991			

Test Set

Label	Precision	Recall	f1	#
0	1	1	1	1169
1	0.8	0.8	0.8	10
ROC/AUC	0.8721			

In tables 4.4.2 we see a comparison pair to pair between the validation set and the test set. We see that the classification for the class-0 (non-gravitational waves) has a perfect score of 1 in every column, while the class-1 (gravitational waves / "Chirps") have a high score (0.8-0.95) in precision, recall and f1 columns. This seems to suggest that the model has a harder time identifying the

chirps. We see, that the ROC/AUC metric induces us to think that the coverage of the cases is quite good as well.

Figure 4.4.3 - The trained LSTM Loss

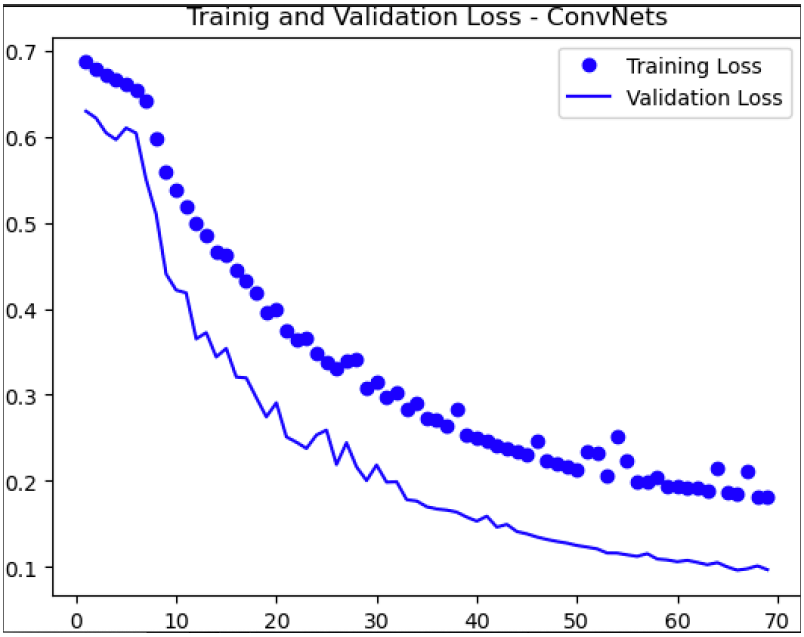
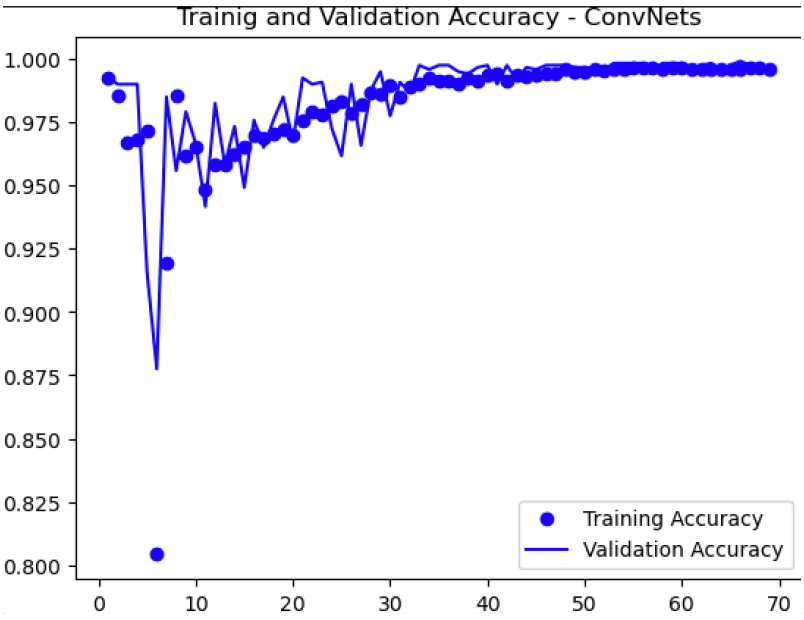


Figure 4.4.4 - The trained LSTM Accuracy



The figures in figure 4.4.3 and 4.4.4, suggests that the risk of overfitting is low, since the model seems to converge in the training and validation sets. The Recurrent Network was able to classify the first class partially. Nevertheless, the ROC/AUC metrics show that we got some false-positives (20% as we can check in the precision for class 1 in table 4.4.2). The model classified the majority of the gravitational waves, introducing some additional images false-positives.

Table 4.4.5 - The prediction made for every set

	No-GW	GW	GW - Identified
Validation Set	1192	9	10
Test Set	1169	10	10

Table 4.4.5 shows that all the chirps were correctly identified; with some false positives.

Table 4.4.6 - The times for the optimal configuration

Convolution Neural Network	Training Times	Predict Times
Train Set	7 minutes	NA
Test Set	NA	1 sec.

The total amount to train and to predict, as shown in table 4.4.6, has reached one of the best times from the four analysed algorithms.

A recurrent network, like the one used here, provides us with a third leg to where to sustent some of our future conclusions. We see here that the spectrograms, in spite of being images, retained its time series nature. Which opens the door for the idea that we can use some of these techniques for the pre classification of the raw data extracted directly from the interferometer's used in LIGO.

4.5 Light GBM Approach

Hardware and configurations pre-sets

This work has been performed over a home hosted virtual machine with the following configuration:

- 2 Cores
- 16 GB of RAM
- Linux Ubuntu 19.0 (workstation) as Operating System

Algorithm's models and evaluated results

After analysing the dataset and understanding the images, we conjecture that the present dataset could be approached in a very systematical form. In this regards, we planned to test the performance of an algorithm's model that is typically more suited for very structured data: a kind of decision trees algorithm. For this approach, we are going to use "Light GBM".

The first model configuration asserted was the default for the lightgbm[51] library, as is out of the box. The default hyper parameters were:

- boosting_type = gbdt - Traditional Gradient Boosting Tree
- num_leaves = 31 - Number of leaves
- max_depth = -1 - Maximum allowed deep
- learning_rate = 0.1 - learning speed
- n_estimators = 100 - Number of boosted trees to fit
- subsample_for_bin = 200000 - number of samples for constructing bins
- objective = Binary - the kind of classifier's "optimizer" function.
- class_weight = None - How much inference has every class type to predict.
- min_split_gain = 0 - Minimum loss reduction required to make a further partition on a leaf node of the tree.
- min_child_weight = 1e-3 – Minimum sum of instance weight (hessian) needed in a child (leaf).
- min_child_samples = 20 – Minimum number of data needed in a child (leaf).
- subsample = 1 – Subsample ratio of the training instance.
- subsample_freq = 0 – Frequence of subsample, <=0 means no enable.
- colsample_bytree = 1 – Subsample ratio of columns when constructing each tree.
- reg_alpha = 0 – L1 regularization term on weights.

- reg_lambda = 0 – L2 regularization term on weights.
- random_state = None – Random number seed.
- n_jobs = -1 – Number of parallel threads.
- silent = True – Whether to print messages while running boosting.
- importance_type ='split' – The type of feature importance to be filled into feature_importances_. If 'split', result contains numbers of times the feature is used in a model.

Table 4.5.1 - The light GBM default model training results

Label	Precision	Recall	f1	#
0	1	1	1	1191
1	0.75	0.67	0.71	9
ROC/AUC	0.9991			

In the table 4.5.1 we see that the ROC/AUC metric, which measures the relation of true positives predicted, gave a very high score of above 0.99. The precision, recall and f1 showed values almost averaging 0.67 to 0.75. Those are high values, which could suggest that the model is classifying the set pretty well. As we can see in Table 4.5.2, the training times were more than acceptable in comparison with the times of the other experiments explored in this paper.

Table 4.5.2 - The times for the optimal configuration

Convolution Neural Network	Training Times	Predict Times
Train Set	5 minutes	NA
Cross Validation	13 minutes	NA
Test Set	NA	1 sec.

Table 4.5.3 - The prediction made for every set

	No-GW	GW	GW - Identified
Validation Set	1192	9	8
Test Set	1169	10	9

As we see in table 4.5.3, the model is capable of classifying almost all the gravitational waves found in the dataset. However, after careful analysis of the classified images, we've detected that the metrics remained high in spite of detecting some false-positives and one less chirp in every case.

We attributed this behaviour to the fact that the basic hyper parameter configuration is placed for multi class and the class weight is ignored. We designed a second model, in which we took special care to reflect the complexities this dataset has, as such:

- objective = "binary" - we set this "explicitly".
- class_weight = "balanced" - which means it will give the due weight to each class in relation to its frequency.

Besides this, as we see in table 4.5.2 the training and fit times are more than acceptable for testing of different models in short spams of time.

Table 4.5.4 - The light GBM balanced model training results

Label	Precision	Recall	f1	#
0	1	1	1	1191
1	1	0.89	0.94	9
ROC/AUC	0.9860			

Table 4.5.5 - The times for the light GBM balanced model configuration

Convolution Neural Network	Training Times	Predict Times
Train Set	5 minutes	NA
Cross Validation	16 minutes	NA
Test Set	NA	1 sec.

Table 4.5.6 - The prediction made for every set for light GBM balanced model

	No-GW	GW	GW - Identified
Validation Set	1192	9	8
Test Set	1169	10	9

As we see in table 4.5.4 some metrics improved against the “default model”; but after analysing the classified images we confirmed that the classification missed one signal in every case (see table 4.5.6), but didn’t produced any false positives (it improved, slightly in comparison with the original model). We tried with other two models that balanced the two classes. As shown in table 4.5.5, the training times and predict performance remained more or less the same.

The third model simply took the 41 gravitational waves from the training set and added an equal amount of shuffled “other” spectrograms, for a total training set of 82 images.

Table 4.5.7 - The light GBM third model training results

Label	Precision	Recall	f1	#
0	1	1	1	1191
1	1	0.89	0.94	9
ROC/AUC	0.9879			

Table 4.5.8- The times for the light GBM third model configuration

Convolution Neural Network	Training Times	Predict Times
Train Set	4 secs	NA
Cross Validation	16 minutes	NA
Test Set	NA	1 sec.

Table 4.5.9 - The prediction made for every set for light GBM third model

	No-GW	GW	GW - Identified
Validation Set	1192	9	118
Test Set	1169	10	135

In spite the fact that table 4.5.7 shows most metrics to fare pretty high (above 0.90) for most except for the f1, the analysis over the identified images is conclusive: this model increases the amount of false positives identified significantly (we can see this, in table 4.5.9). The times for training, as we can check in table 4.5.8 are more or less the same ones that we saw before. Our conjecture is that the set has a low amount of class-0 during training to be able to generalise the model well.

We tested a fourth model, that repeated the 41 “chirps” until it reached the amount of “other signals”, doubling the size of the dataset. This will take into account the whole set of available

signals involved and the ones we already know as“gravitational waves” to an artificially balanced set of more than 10.000 images. The idea here, was to balance the class-1 and class-0 before training.

Table 4.5.7 - The light GBM third model training results

Label	Precision	Recall	f1	#
0	1	1	1	1191
1	0.80	0.89	0.84	9
ROC/AUC	0.9475			

Table 4.5.8- The times for the light GBM third model configuration

Convolution Neural Network	Training Times	Predict Times
Train Set	7 minutes	NA
Cross Validation	16 minutes	NA
Test Set	NA	1 sec.

Table 4.5.9 - The prediction made for every set for light GBM third model

	No-GW	GW	GW - Identified
Validation Set	1192	9	10
Test Set	1169	10	10

As we may observe in table 4.5.7 and 4.5.9 this model arrangement was the one with the best performance in terms of training times and identification of “chirps”. The amount of available gravitational waves in the set and its regularity at the time of identification of them seems to be the key issue with the efforts that these models try to convey. We used a “decision tree algorithm”, thought to deal with more structured datasets than images. Even thought, was capable of classifying most of the images in the validation and test dataset right after balancing the amount of data searched. Besides this, in comparison with the former models, the training times were more or less the same for every tested model (see table 4.5.8).

In this algorithm, we choose to test four different models, and described the whole set of results we achieved with different approaches. This helped us to understand:

- That the whole dataset, as spectrograms were, is very regular. Such as it can be correctly classified by a decision tree kind of algorithm.

- For any given model the imbalance in the classes affected them beyond what balancing capabilities could be provided by the different libraries. It is key, that the analyst takes his time to imbue the dataset and the model with his domain knowledge.

5. Conclusions

Let's start with a comparison of the metrics, side by side, and by algorithm, for the best models trained for every approach.

Table 5.1 - The metrics compared for every model against the Test set

Class-1 prediction	Gravity Spy	Linear SVC	CNN	RNN	Light GBM
Precission	1	1	1	0.80	0.80
Recall	1	1	0.90	0.80	0.89
f1	NA	1	0.95	0.80	0.84
ROC/AUC	1 (approx.)*	1	1	0.87	0.94

*(In regards of table's 5.1 ROC/AUC metric for the Gravity Spy column; it was inferred from table 4.1.1.3.)

We see in table 5.1, that Linear SVC got the most approximated results to the ones obtained in the Gravity Spy paper. All the other algorithms aided to get a better understanding of the dataset nature. One additional important remark is this: the Gravity Spy's paper achieved these results using a "Deep Learning" algorithm.

Table 5.2 - The training times compared across algorithms for the Test set

	Gravity Spy	Linear SVC	CNN	RNN	Light GBM
Training times	Unknown	76 secs	20 minutes	7 minutes	7 minutes

As we can see in table 5.2, the Linear SVC training times are very low in comparison with all the others (we do not know the times for Gravity Spy algorithm, but we can guess they could be in the range of the CNN and RNN). For the performance shown in table 5.1 these numbers, for the Linear SVC, seems to be very good. Taking into account that, strictly speaking, these results aren't directly comparable to the ones obtained in the Gravity Spy paper since they trained a CNN over different times spans (the four of it: 0.5, 1, 1.5 and 2 seconds). Still, the results remains very relevant in the face of understanding the challenges ahead and the dataset itself. When not, to give a glance over the limits that these algorithms could present within the presented models in the process of signal classification throughout spectrograms.

We saw the performance of several algorithms, some from the body of knowledge of machine learning and others from the side of deep learning. We saw algorithms oriented to classify images that performed poorly or were more difficult to configure than more structured ones, and we saw algorithms that where supposed to fare pretty bad with un-structured data and performed well

still with this dataset. As we explained in **appendix A**, this is a particular, very well categorised dataset. Most of the images are centred, the 140x170 matrix is regular and standardised, the time series complexity was constrained to a span of half a second and we worked over it to approach the same results that the gravity spy paper has obtained. We were, in essence, trespassing "well known ground".

The linear SVC algorithm, in terms of training times, model complexity, and general metrics got the best performance in all the relevant dimensions.

Some of the questions we intended to respond, got the following answers:

- **How hard might be for some of the presented algorithms to identify a gravitational wave given its due spectrogram?**

The SVMs where the best algorithm of the few we tested, having the lowest threshold for configuration, training and classification. The only caveat could be how much these metrics suppose to provide a model that has overfitted the dataset. And more importantly, if such a model could be generalised for classification of the regular signals obtained from an interferometer.

- **Can we approximate the classification of the same dataset, when it comes to "chirps" to what the Gravity Spy has achieved?**

As we saw in figure 4.1.1.3, the "Chirp" has ben classified with precision, recall and other metrics with a "perfect" score of one. Some of our algorithms draw it close to it, and the SVM got it as well. However, we tried to increase a little bit of the study done for the original work, adding some other metrics, like ROC/AUC. We checked that the overall performance was pretty high as well.

- **Can we validate alternative ways to classify "chirps" that weren't present in the original paper?**

We tested at least two different algorithms that weren't in the original paper and fared pretty well in binary-classification.

- **Can we be able to train a model that classifies at least one "Chirp"?**

Yes, we did it.

- **It is the same to classify the gravity spy's dataset with a deep learning technique than with a more traditional machine learning technique?**

No, it is not. The complexity to design a deep learning network, isn't justified by its overall performance in comparison with the mode "classical" machine learning algorithms.

- **These spectrograms supposes a un-structured dataset, given that they are images, or its data is truly regular?**

We saw that algorithms aimed to classify more structured datasets, like data tables, fared pretty high in their metrics. Giving us an insight to the nature of the dataset. These waves, in spite of being flooded by noise, seems to be very regular. This gives us the idea that the scientist behind their analysis could be more concerned about how to deal with "glitches" and their effect over new discoveries than how to identify well known signals (Even if they came along in a "messy" sample).

However, in spite this, these analysis were performed over a “controlled” dataset. The real dataset is massive, and its processing is challenging even for well known signal’s forms.

- **Can we arrive to some of the same conclusions or validate some of the conclusions obtained by the gravity spy paper?**

The original paper was a work of engineering; it was looking for automatic classification of most of the know “glitches” that can be found in a spectrogram by an astrophysicist. As an appliance, the models found in this paper could be trained to classify every glitch against all others using a binary classifier. Identifying each single kind of glitch, when they appear. Therefore, opening the door to a different approach: classifying specific events instead of all of them at once. The benefit of this approach is that, in a prospective “future” tool, the astrophysicist could try to “pick” a specific model to identify each type of specific event that he might like to pick. Thinking in a “software tool”, this can be an advantage. I imagine this tool counting with a classifier for every possible celestial event acting as a “plugin” of this software tool, aiding in the visualisation of the signals. Allowing the astrophysicists to pick and choose which model he will use, to try to filter a specific kind of event, glitch or error. Or all of them, at once, in “parallel” instead of in “series”. These are just speculations, and should be explored in a different work.

This research, was just the kick-off for a deeper future analysis over these kind of signals, providing just another point of view for the same phenomena.

6. Appendix A - the dataset in detail

The dataset is the same one mentioned in the "Gravity Spy" paper that serves as base for this research, which was publicly available in Zenodo's site[52]. We used for this work a subset of it composed of two files:

- "trainingse_v1d1_metadata.csv": a comma separated file composed by the hashed GPS times of every observation, its classification. This file has many columns, but there are a few which are the main ones: gravityspy_id, label, and sample_type.
    - The *gravityspy_id* is the unique 10 character hash given to every "Gravity Spy" sample.
    - The *label* is the string label of the sample (its classification).
    - The *sample_type* indicates whether this sample was used in the paper for testing, training or validating the models.
- "trainingset1v1d1.h5": an .h5/HDF format for hierarchical multidimensional scientific data storage[53]. This file is composed by, approximately, 40,000 rows in 3.1 Gb. Each one of these with a 140x170 matricidal representation of every signal's spectrogram organised in the following hierarchical way:
    - label -> sample_type -> gravityspy_id -> "{time range}.png"

("Time range" being: "0.5", "1.0", "2.0" or "4.0")

Given the scope of this work, we extracted the spectrograms for the "half second" observations and mapped them all to each row in the .csv file; pairing the image's matrices with the 30's fields and keeping just the following columns:

- **label**: Air compressor, Blip, Chirp, Extremely loud, Helix, Koi fish, Light modulation, Low frequency burst, Low frequency line, None of the above, No glitch, Paired doves, Power line , Repeating blips, Scattered light, Scratchy, Tomte, Violin mode harmonic, Wandering line, Whistle
- **sample_type**: train, test or validation
- **png**: the 140x170 float matrix that represents the spectrogram of a single observation.

Since these images are "glitchs" classifications, we do know that there are some that resemble "gravitational waves". As far as we could tell, these are the ones catalogued as "Chirps". They are roughly the 0.8% of the dataset. We changed there labels to work as such:

- **Label**: 1 and 0; where 1 represents "Gravitational Wave" and 0 "Non-Gravitational Wave".

We end up with a pandas' data frame composed of these three columns. Which includes the spectrograms of "candidate" gravitational waves.

One remarkable note about the 140x170 matrices: these are representations of an internal *mathplotlib*[54] compatible format. In this format, which entails the usage of the function *imshow*[55], we can store and retrieve the main spectrogram's image in a float matrix of 140x170. Such a matrix

can be represented by this library through a colormap. These color maps represents the RGB pixel color range with a single float number between 0 and 1[56].

Summary of the final dataset with which all training and classifications will be made:

- **Label**: 1 and 0, where 1 represents “Gravitational Wave” and 0 “Non-Gravitational Wave“.
- **sample_type**: train, test or validation
- **png**: the 140x170 float matrix that represents the spectrogram of a single observation that has a color value that goes from 0 to 1.

The Python script that attains this, could be found in https://github.com/exemartinez/gravitational_waves_classifiers/blob/master/gw/generate_gw_dataset.py

We stored this “tailored” dataset into a .pickle file for later use. Therefore, we performed other transformations in order to work with the different libraries and adapt the sets to their required input’s format.

The whole set is organised in the following way:

- Training set: 5587 images. (41 “chirps” waves)
- Testing set: 1179 images (10 “chirps” waves)
- Validation set: 1200 images (9 “chirps” waves)

7. Appendix B - The metrics used in this work explained

All the metrics used in this paper got a mark that goes from zero to one; zero being the worse possible escenario and one the highest "perfect" rank[57].

Accuracy[58]

This metric creates two local variables, total and count that are used to compute the frequency with which predicted labels matches true labels. Since the kind of classification we've used is binary classification, all the models used the Keras "binary_accuracy" keyword. This metric is the default for Deep Learning networks in the Keras / Tensorflow library. When not, we used Root Mean Squared Error[59].

Loss[60]

The purpose of loss functions is to compute the quantity that a model should seek to minimise during training. We can write our own loss functions, which will compute a value between 0 and 1, the lowest the value, the better. There are many loss functions which outputs a "loss" value. The one we used the most for our models is Binary Cross Entropy[61], as it is suggested in the Keras library. This metric is the default for Deep Learning networks in the Keras / Tensorflow library.

ROC / AUC

Receiver Operating Characteristic / Area Under the Curve, is the area under the ROC curve, which represents the diagnostic skill of a binary classifier system. The ROC curve is created by plotting the true positive rate against the false positive rate. The higher the score, the better.

Precision

The precision is the amount of "correctly" identified positive results divided by the number of all possible positive results, including those one which were not identified correctly. In essence:

True Positives / (All Real Positives) + (False positives)

The higher the score, the better.

Recall

The recall is the number of correctly identified positive results divided by the amount of all samples that should have been identified as positive. In essence:

True Positives / All Real Positives

The higher the score, the better.

F1

The F1 score (also F-score or F-measure) is a measure of accuracy. It is calculated from the precision and recall of the test. In essence:

$$\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

The higher the score, the better.

8. Acknowledgements

We place special thanks for this work to:

To Rodrigo Ramele from Instituto Tecnologico de Buenos Aires (ITBA), for being my caring tutor and answering the most silly questions at the most unholy hours.

To Brina Martinez and the University of Texas (CGWA) for their support and goodwill. She helped us to understand the complexities of the whole subject, reviewed this paper's proposal and was of terrific help trying to give us a glance over deep space physics.

To Victor De los Santos and the South Texas Astronomical Society (STAS), who brought this project to my table and provided all the relevant contacts, links and the support to continue in the dearest times.

To Jonah B. Kanner and the California Institute of Technology (CALTECH), for opening they dataset and allowing us to play with it months before we started to look after a classification strategy. But more than anything, for highlighting us the existence of this dataset and their related papers.

To Eric Chassan de Mottin from LIGO, for helping me to understand the complexities of the gravitational waves subject, the challenges attached to the domain and for pointing out the right exit to all the due challenges.

To my dear wife and children, for given me the space to look after my passions, in spite taking away shared time with them.

Science it's a shared field full of passion and collaboration, nothing could be achieved if we weren't at the shoulders of titans.

9. References

[1] Caltech, MIT, NSF (2020), "What is LIGO" [LIGO official site] recovered from https://www.ligo.caltech.edu/page/what-is-ligo

[2] Caltech, MIT, NSF (2020), "What is LIGO" [LIGO official site] recovered from https://www.ligo.caltech.edu/page/what-is-ligo

[3] NSF, LIGO, INFN, CNRS, LSC, EGO, VIRGO (2020). "Gravitational Wave Open Science Center" [Official site explaining how their interferometers works] recovered from https://www.gw-openscience.org/about/

[4] Caltech, MIT, NSF (2020), "What is an interferometer", recovered from https://www.ligo.caltech.edu/page/what-is-interferometer

[5] NASA (2020), "What is a gravitational Wave?" [NASA's official Site] recovered from https://spaceplace.nasa.gov/gravitational-waves/en/

[6] NASA (2020), "What is a gravitational Wave?" [NASA's official Site] recovered from https://spaceplace.nasa.gov/gravitational-waves/en/

[7] N. Mavalvala, "Precision measurement at the quantum limit in gravitational wave detectors," *2014 Conference on Lasers and Electro-Optics (CLEO) - Laser Science to Photonic Applications*, San Jose, CA, 2014, pp. 1-2.

[8] NSF, LIGO, INFN, CNRS, LSC, EGO, VIRGO (2020). (Datasets and coding libraries). Recovered from https://www.gw-openscience.org/archive/O2_4KHZ_R1/

[9] NSF, LIGO, INFN, CNRS, LSC, EGO, VIRGO (2020). (Datasets and coding libraries). Recovered from https://www.gw-openscience.org/yellow_box/

[10] "Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science" M Zevin1, S Coughlin1, S Bahaadini2, E Besler2, N Rohani2, S Allen3, M Cabero4, K Crowston5, A K Katsaggelos2, S L Larson1,3, T K Lee6, C Lintott7, T B Littenberg8, A Lundgren4, C Østerlund5, J R Smith9, L Trouille1,3 and V Kalogera1Hide (February, 2017) doi: https://doi.org/10.1088/1361-6382/aa5cea

[11] Daniel George, E.A. Huerta, (2017) "Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data", doi: 10.1016/j.physletb.2017.12.053

[12] "Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science" M Zevin1, S Coughlin1, S Bahaadini2, E Besler2, N Rohani2, S Allen3, M Cabero4, K Crowston5, A K Katsaggelos2, S L Larson1,3, T K Lee6, C Lintott7, T B Littenberg8, A Lundgren4, C Østerlund5, J R Smith9, L Trouille1,3 and V Kalogera1Hide (February, 2017) doi: https://doi.org/10.1088/1361-6382/aa5cea

[13] Zooniverse (2020), "Help scientists at LIGO search for gravitational waves, the elusive ripples of spacetime.". Recovered from https://www.zooniverse.org/projects/zooniverse/gravity-spy

[14] GWOSC (2020), "Gravitational-Wave Open Data Workshop #3" [dataset and coding libraries], recovered fromhttps://github.com/gw-odw/odw-2019/blob/master/Challenge/CHALLENGE.md

[15] GWOSC (2020), "Gravitational-Wave Open Data Workshop #3" [Introduction to the workshop], recovered from https://www.gw-openscience.org/s/workshop3/

[16] Vallisneri, Michele et al. (2015), "The LIGO Open Science Center." Journal of Physics: Conference Series 610 - 2015: 012021. DOI: 10.1088/1742-6596/610/1/012021

[17] Kuroda, Kazuaki, Wei-Tou Ni, and Wei-Ping Pan. (2015), "Gravitational Waves: Classification, Methods of Detection, Sensitivities and Sources." International Journal of Modern Physics D 24.14: 1530031. DOI: 10.1142/S0218271815300311

[18] Daniel George, E.A. Huerta, (2017) "Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data", doi: 10.1016/j.physletb.2017.12.053

[19] LIGO (2017), "LIGO Data Management Plan, June 2017", [Data release plan documentation], recovered from: https://dcc.ligo.org/public/0009/M1000066/025/LIGO-M1000066-v25.pdf

[20] LIGO (2017), "LIGO Data Management Plan, June 2017", page 10, [Data release plan documentation], recovered from: https://dcc.ligo.org/public/0009/M1000066/025/LIGO-M1000066-v25.pdf

[21] M Zevin1, S Coughlin1, S Bahaadini2, E Besler2, N Rohani2, S Allen3, M Cabero4, K Crowston5, A K Katsaggelos2, S L Larson1,3, T K Lee6, C Lintott7, T B Littenberg8, A Lundgren4, C Østerlund5, J R Smith9, L Trouille1,3 and V Kalogera1Hide, "Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science", (February, 2017) doi: https://doi.org/10.1088/1361-6382/aa5cea

[22] Mack, Wolfgang, and Emanuel A. P. Habets. (2019) "Deep Filtering: Signal Extraction and Reconstruction Using Complex Time-Frequency Filters.", doi: 10.1109/LSP.2019.2955818

[23] Daniel George, E.A. Huerta, (2017) "Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data", doi: 10.1016/j.physletb.2017.12.053

[24] Sci Kit Learn (Version 0.23) [Software Library], recovered from https://scikit-learn.org/

[25] Pandas (Version 1.0.3), recovered from https://pandas.pydata.org/docs/

[26] Keras (Version 2.0) [software library] recovered from https://keras.io/

[27] Numpy (Version 1.18) [Software library] recovered from https://numpy.org/

[28] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R. New York: Springer.

[29] Chollet, F. (2018). *Deep learning with Python*. Shelter Island, NY: Manning Publications,

[30] Chollet, F. (2018). *Deep learning with Python*. Shelter Island, NY: Manning Publications.

[31] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R. New York: Springer.

[32] "Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science" M Zevin1, S Coughlin1, S Bahaadini2, E Besler2, N Rohani2, S Allen3, M Cabero4, K Crowston5, A K Katsaggelos2, S L Larson1,3, T K Lee6, C Lintott7, T B Littenberg8, A Lundgren4, C Østerlund5, J R Smith9, L Trouille1,3 and V Kalogera1Hide (February, 2017) doi: https://doi.org/10.1088/1361-6382/aa5cea

[33] Zenodo(2018), "Updated Gravity Spy Data Set" [Dataset and coding libraries] recovered from: https://zenodo.org/record/1476551#.Xu6TaZNKjNw, DOI: 10.5281/zenodo.1476551

[34] "Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science" M Zevin1, S Coughlin1, S Bahaadini2, E Besler2, N Rohani2, S Allen3, M Cabero4, K Crowston5, A K Katsaggelos2, S L Larson1,3, T K Lee6, C Lintott7, T B Littenberg8, A Lundgren4, C Østerlund5, J R Smith9, L Trouille1,3 and V Kalogera1Hide (February, 2017), page 5, figure 1, doi: https://doi.org/10.1088/1361-6382/aa5cea

[35]James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R. New York: Springer, pag. 145.

[36]"Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science" M Zevin1, S Coughlin1, S Bahaadini2, E Besler2, N Rohani2, S Allen3, M Cabero4, K Crowston5, A K Katsaggelos2, S L Larson1,3, T K Lee6, C Lintott7, T B Littenberg8, A Lundgren4, C Østerlund5, J R Smith9, L Trouille1,3 and V Kalogera1Hide (February, 2017), page 18, figure 7, doi: https://doi.org/10.1088/1361-6382/aa5cea

[37] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R. New York: Springer, pag. 344.

[38] SciKit Lean (version 0.23.1) [Software library] recovered from: https://scikit-learn.org/stable/

[39]James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R. New York: Springer, chapter 5.

[40] Source code (Unique version), from our own production: https://github.com/exemartinez/gravitational_waves_classifiers/blob/master/gw/predict_linearsvc_gw.py

[41] RABINOVICH, S. G. (2018). EVALUATING MEASUREMENT ACCURACY: A practical approach. Place of publication not identified: SPRINGER.

[42] Source code (Unique version), from our own production: https://github.com/exemartinez/gravitational_waves_classifiers/blob/master/gw/train_linear_svc_model.py

[43] Source code (Unique version), from our own production: https://github.com/exemartinez/gravitational_waves_classifiers/blob/master/gw/predict_linearsvc_gw.py

[44] Microsoft's Azure Open Datasets(2020), "Handwritten Digits" [Dataset and coding libraries] recovered from: https://azure.microsoft.com/es-mx/services/open-datasets/catalog/mnist/

[45] Menshawy, A. (2018). Deep learning by example: A hands-on guide to implementing advanced machine learning algorithms and neural networks. Birmingham: Packt Publishing.

[46] Chollet, F. (2018). *Deep learning with Python*. Shelter Island, NY: Manning Publications, pag. 120-122

[47]Chollet, F. (2018). *Deep learning with Python*. Shelter Island, NY: Manning Publications, pag. 133-138

[48]Chollet, F. (2018). *Deep learning with Python*. Shelter Island, NY: Manning Publications, pag. 202-206

[49] Hyndman, Rob J.; Koehler, Anne B. (2006). "Another look at measures of forecast accuracy". *International Journal of Forecasting*. **22** (4): 679–688. doi:10.1016/j.ijforecast.2006.03.001

[50] Murphy, Kevin (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

[51]Light GBM (version 2.3.1) [Software library] recovered from: https://pypi.org/project/lightgbm/

[52] Zenodo(2018), "Updated Gravity Spy Data Set" [Dataset and coding libraries] recovered from: https://zenodo.org/record/1476551#.Xu6TaZNKjNw, DOI: 10.5281/zenodo.1476551

[53] FileInfo (2020), ".H5 file extension", recovered from: https://fileinfo.com/extension/h5

[54] MathPlotLib (Version 3.2.2) [software library] recovered from https://matplotlib.org/

[55] MathPlotLib, "matplotlib.pyplot.imshow" [Dataset and coding libraries], recovered from https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.imshow.html

[56]MathPlotLib, "matplotlib.colors.colormap" [Dataset and coding libraries], recovered from https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.colors.Colormap.html#matplotlib.colors.Colormap

[57]James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R. New York: Springer.

[58] Goodfellow, I., Bengio, Y., Courville, A. (2018). Deep Learning. Frechen: MITP, pages 103-14.

[59] Hyndman, Rob J.; Koehler, Anne B. (2006). "Another look at measures of forecast accuracy". *International Journal of Forecasting*. **22** (4): 679–688. doi:10.1016/j.ijforecast.2006.03.001

[60] Goodfellow, I., Bengio, Y., Courville, A. (2018). Deep Learning. Frechen: MITP, pages 82-93.

[61] Murphy, Kevin (2012). Machine Learning: A Probabilistic Perspective. MIT Press.