CARRERA: ESPECIALIZACIÓN EN CIENCIA DE DATOS

TALLER: 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

Profesores:

Dra. Alicia Mon

Lugar y Fecha: Buenos Aires, 04 de Junio de 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 classificatory 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 from the knowledge base of machine learning and deep learning. 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 engine for gravitational waves signal's classification.

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.

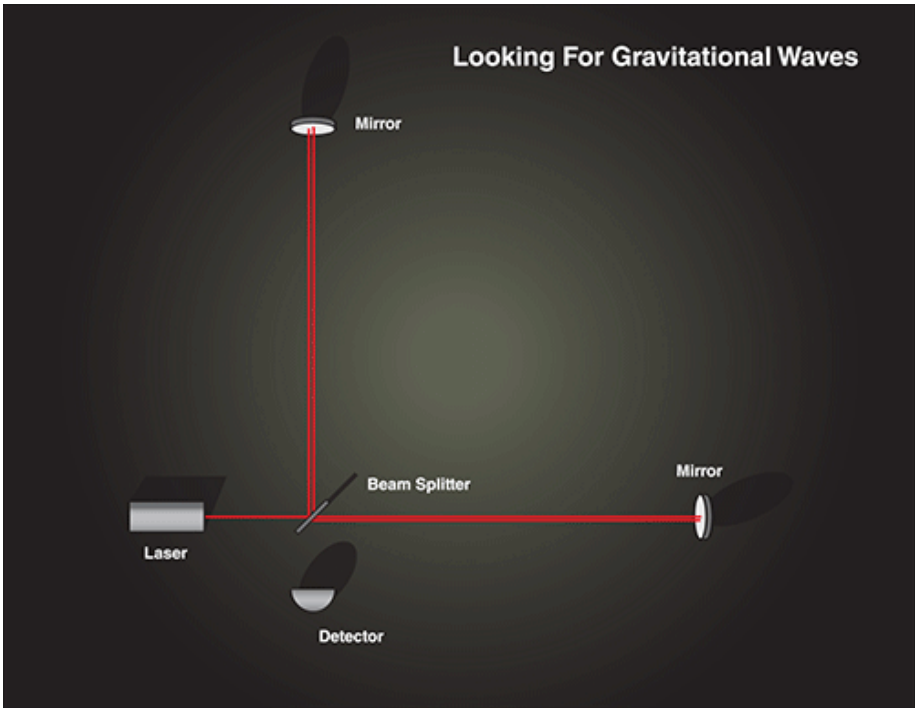
Gravitational waves are signals emitted by objects of high mass that resides in deep space. Usually, they can be emitted in the occurrence of a celestial event. They constitute disturbances in the space-time fabric. These object are typically massive and of the size of a star. These signals occurs in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

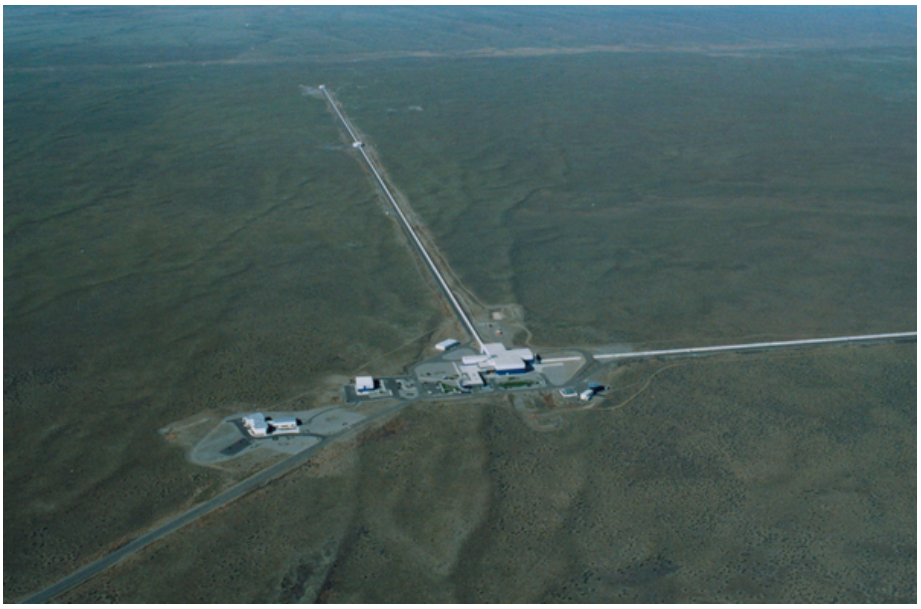
The instruments that take those signals from deep space are called interferometers[4] and work inside the LIGO observatories. An "interferometer" it is analogous to a telescope for electromagnetic signals. As the name states, is a sensor designed to identify interferences. There are of several sizes, but the ones you can find in Livingston and Handford observatories are particularly big. Size matters, and their intent is to catch the specific interferences produced by a gravitational event 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 these measurements. One of the few being the signal's filtering and cleanse[7]. The signals taken by the sensors are digitalised for its later analysis. They are structured in "data-burst" that are measured by second[8].

Some of the historical data is available and refers to past observation's run, but 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 standarized and resolved in structured datasets that can be consumed by a regular Python script. There are in place several efforts to work in the measurements and clean 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 are going to be centering our efforts to 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.

In spite 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 a gravitational wave.

- 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:

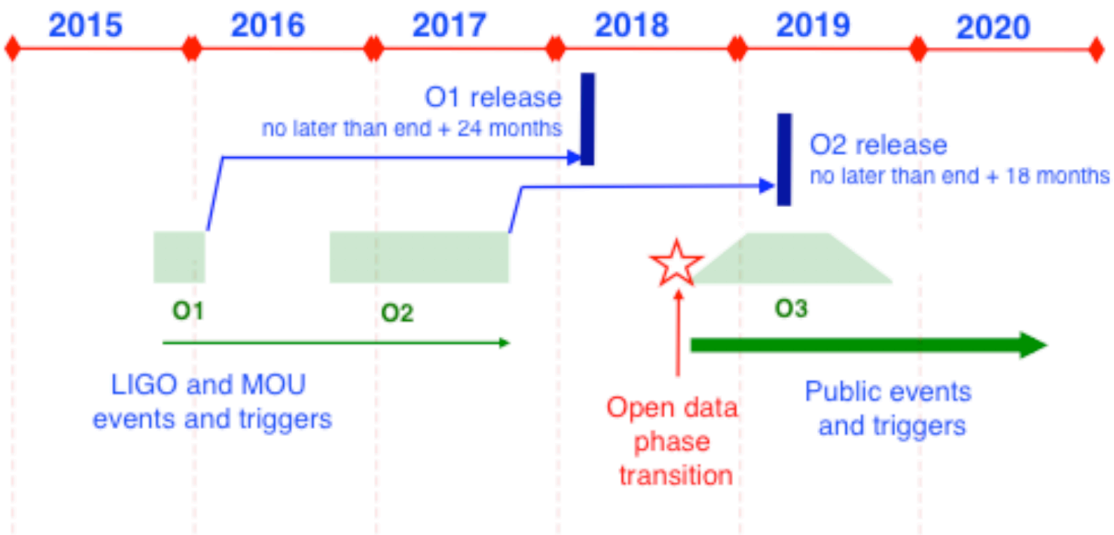
- Provide a comparison of several models' performance to bring more light over the classification and characterisation for noisy signals when represented as spectrograms.
- It will provide several models capable of classifying a spectrogram in a binary way: either a candidate gravitational wave or not.
- 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 as well 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 signal 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 astrophycisist.

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. This is the dataset provided by the "Gravity Spy paper". We will to use it for the intent of evaluating four algorithms performance in the classification of the "chirps" saw 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. Study justification~~

~~Application of some of the ideas of machine learning and deep filtering[24] to the identification of the gravitational signals within the scope of the given dataset and evaluate its performance could be of great use for the further development of the field. We understand that this could give us a better understanding of how different techniques could behave in relation with such a complex and noisy signal. If we reach the correct identification of a subset of the events, we'll be placed a first steps into the classification of such signals. Such applications could hit onto the following fields:~~

- ~~Filtering of noise signals obtained from other deep space sources.~~
- ~~Identification of patterns in sensors of other sorts in noisy environments.~~
- ~~Voice filtering in noisy environments.~~

- Surveillance and the development of better sensors in the building of satellites.
- Identification of different human verbal languages and their translations.

These signals provide us a challenge to identify them. Developing or reproducing techniques on such a dataset could help us to apply the same models to different problems where the signals are dim or subceptible to heavy noise, but where we know "what we are looking for".

However, we are not going to classify all the universe of these signals available. We will suscribe our research to the dataset freely available and try to classify the spectrogram of these signals. The value is in the comparison with the results obtained in the Gravity Spy's paper and see if there are differences in the classification performance of two or more constrasted techniques.

5. Study scope

Working with the dataset trainingsetv1d1.h5, available from the site https://zenodo.org/record/1476551#.Xu6TaZNKjNw, we will apply techniques of deep learning in order to identify at least one of the "Chrips" classified in the "validation" set of the 0.5 seconds observation range. This dataset is a real observation obtained during the second observation run denominated O2[25]. The scope of this work is to try to train at least four different machine learning algorithms with the intention of evaluate their performance on the classification of, at least, one gravitational wave. The dataset is fully pre-classified and ready. It has been pre-divided in training, validation and test sets for this work.

6.Hypothesis

The conjecture is that given the pre-classified dataset, we will be able to train at least two machine learning algorithms capable of classifying at least one of the gravitational events introduced (a "Chirp"). For this to happen, we will be able to use the scikitlearn[26], pandas[27], keras[28] and numpy[29] libraries provided by the body of tools found in the Python's universe. The result of this work will be one or more trained models; these will be capable of classifying a spectrogram of a signal into as a gravitational wave with a certain degree of accuracy. The aim of this work is to compare the performance of these models between them and then check it against the benchmark set by the Gravity Spy's paper; by all means, we look after reproducibility and analysis of this specific previous work in just one dimension: to classify a gravitational wave (a chirp) evaluating different algorithms performance within the very same set of hardware arrangement.

7. Objectives

To compare the classification performances of a linear SVC, CNN, RNN and LightGBM algorithms at their machine learning models capable of classifying one or more gravitational signals found in the spectrograms available in the given dataset for the range of 0.5 seconds. The rationale behind it, has to be with reproducibility of some of the work found in Gravity Spy plus a deeper understanding of how these algorithms perform over the given dataset.

This objective was restricted to the data file "trainingsetv1d1.h5" openly found[30] on the Zenodo repositories. Since the timespan we will be operating with will be constrained to 0.5 seconds spectrograms, the classification problem changes to a scope of image classification.

Sub-goals:

- Download the datasets trainingsetv1d1.h5; place them into a spark architecture for its later treatment.
- Train a linear SVC model to work as "Chirp spectrogram classifier". Note its performance and describe the model configuration.
- Train a Recurrent Neural Network model to work as "Chirp spectrogram classifier". Note its performance and describe the model configuration.
- Train a Convolutional Neural Network model to work as "Chirp spectrogram classifier". Note its performance and describe the model configuration.
- Train a lightGBM model to work as "Chirp spectrogram classifier". Note its performance and describe the model configuration.
- Compare the ROC/AUC, Accuracy, Loss and training times for every model; compare these methods to assest their performance against one another.
- Compare it with the best model done by the Zooniverse effort for Gravity Spy and draw conclusions over the classifiers performance over the dataset and the nature of the dataset itself from the point of view of a Data Scientist.

8. 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.

8.1 Methods and Materials

Algorithms and model's sources

For this research we will be able to use the scikitlearn[31], pandas[32], keras[33] and numpy[34] libraries provided by the body of tools found in the Python's universe. We've worked with four different algorithms:

Machine Learning algorithms[35]

- 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. They 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[36]

- 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". They are good for classification of time series data.

We have develop 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 training and validating the dataset are the methods suggested by François Chollet[37] and James Gareth "et al"[38] 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"[39] paper, since the same comes already divided and tagged in these three subsets.)

After this step was taken, we configured the different algorithms, train them, save the model and measure 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 similar “Loss” and “Accuracy” functions in order to estimate how “good” the trained model is. (they are a percentage and therefore could be compared).
- 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.

Research’s experiment’s code

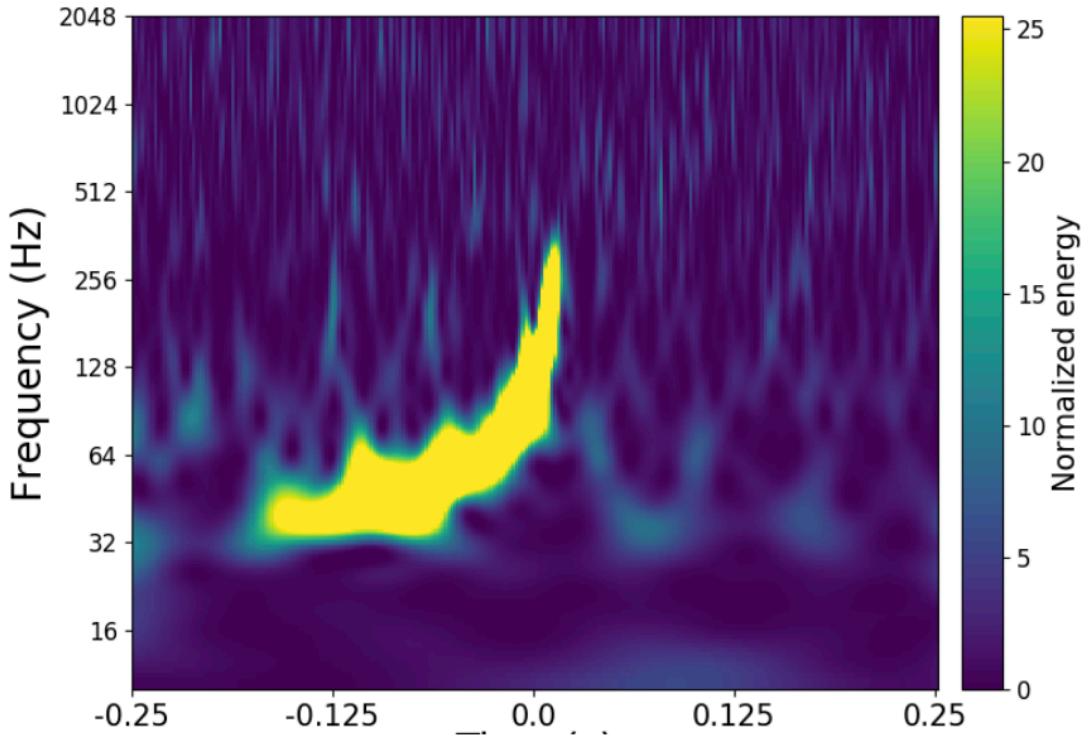
All the algorithms and experiments performed during this research could be found in: https://github.com/exemartinez/gravitational_waves_classifiers/tree/master/gw

8.1.1 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.

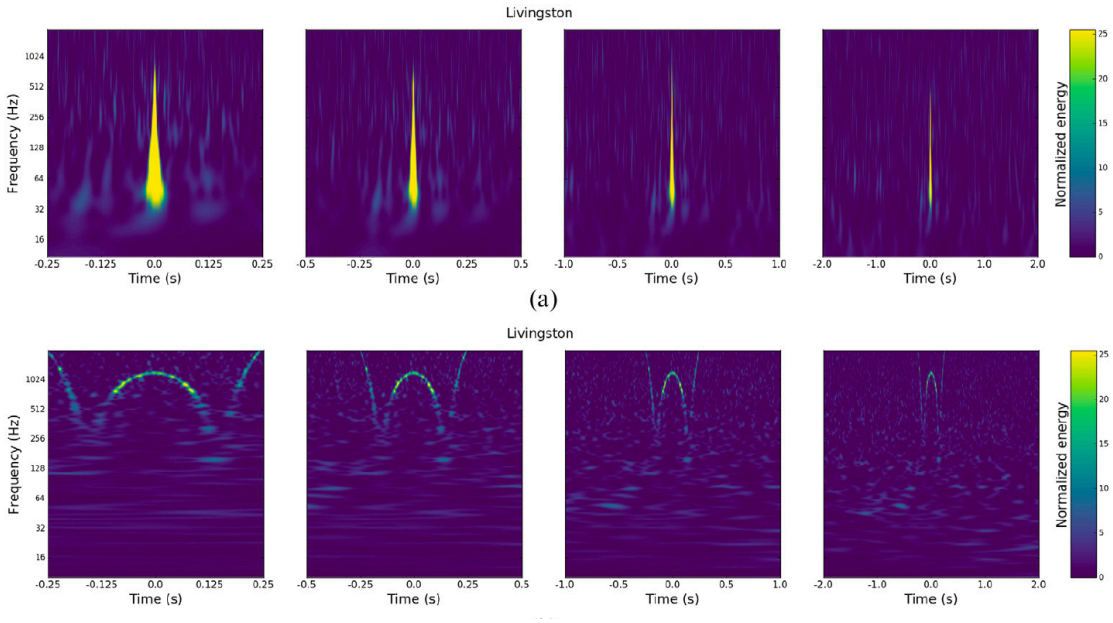
A gravitational wave looks like a “chirp” in the spectrogram, such a signal looks like the one represented in figure 8.1.1.1, which was recovered from the dataset we’ll be dealing with.

Figure 8.1.1.1 - This is a “Chirp”, a signal candidate to be classified as a gravitational wave by a Physicist.[40]



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

Figure 8.1.1.2 - Several spectrograms examples that aren't chirps neither gravitational waves.[41]

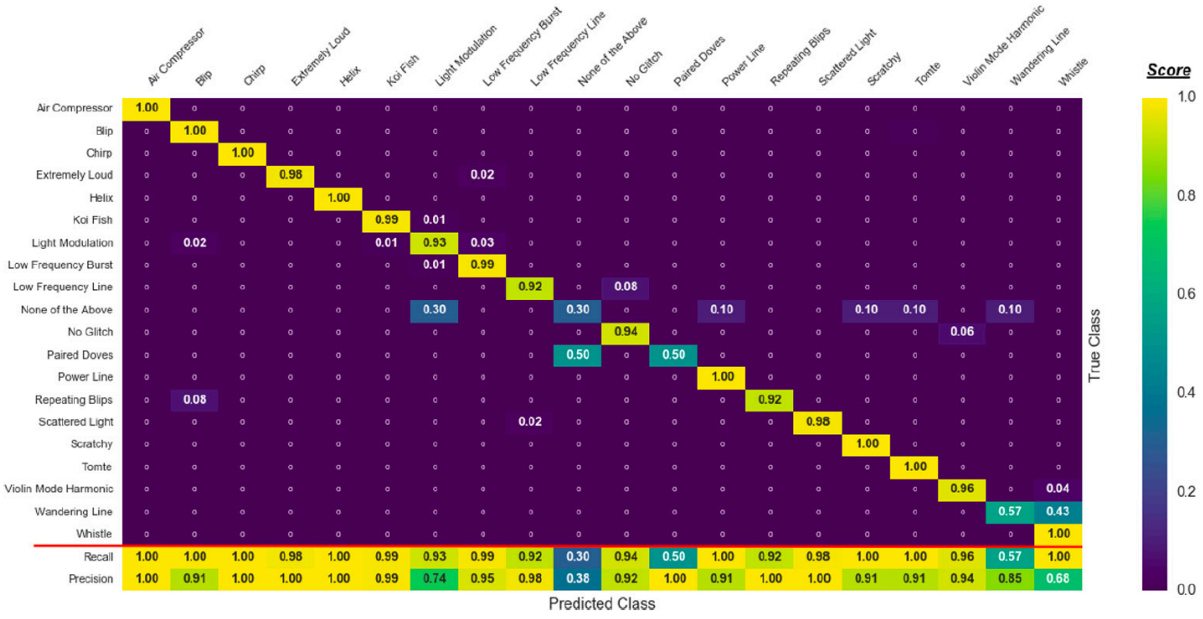


In figure 8.1.1.2 we see the different kind of images and how their timeframes affect their look; we'll deal only with images in the 0.5 seconds spam and analyse the findings.

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 8.1.1.3, where the author sustains the confusion matrix for the CNN classification he made over the same dataset; we'll be doing the same thing in a binary approach.

Figure 8.1.1.3 - The confusion matrix for the classification of the 20 glitches with a trained CNN[42].



8.2 Classification throughout 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

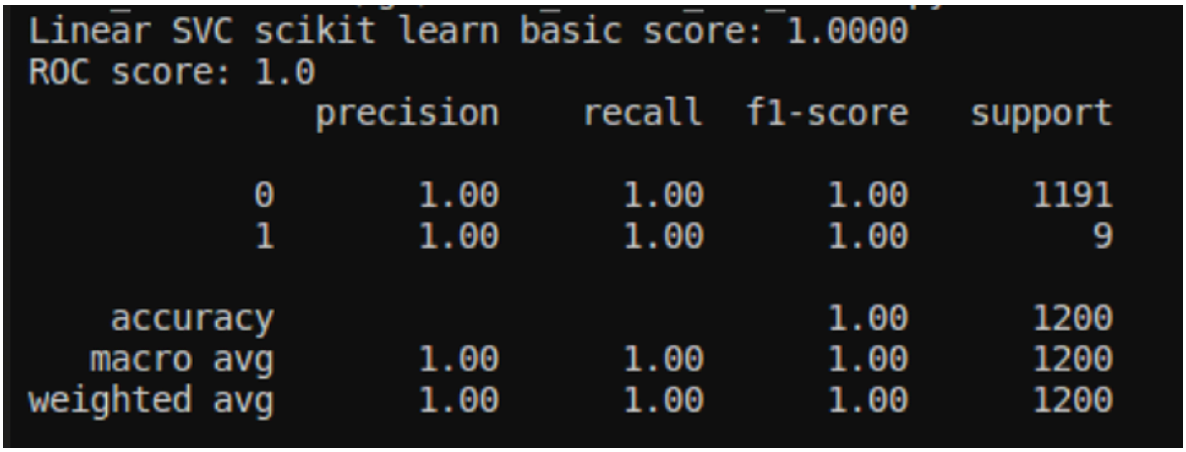
Algorithm's models and evaluated results
The Linear SVC (Linear Support Vector Classifier) it's a kind of support vector machine designed to deal with binary classification of complex data. It is a special kind of support vector machine with a linear kernel and a distance from the hyperplane to classify two classes[43].

We used the Sci Kit Learn library[44] to test this classification and draw metrics from it. The configuration we build our first classification upon is 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 5587 float matrices that corresponds to every spectrogram. We know that in this set we'll face roughly 41 "Chirps".
- We provided a balanced "class_weight"; since we know that the chirps supposes less than the 1% of the observations in that set.

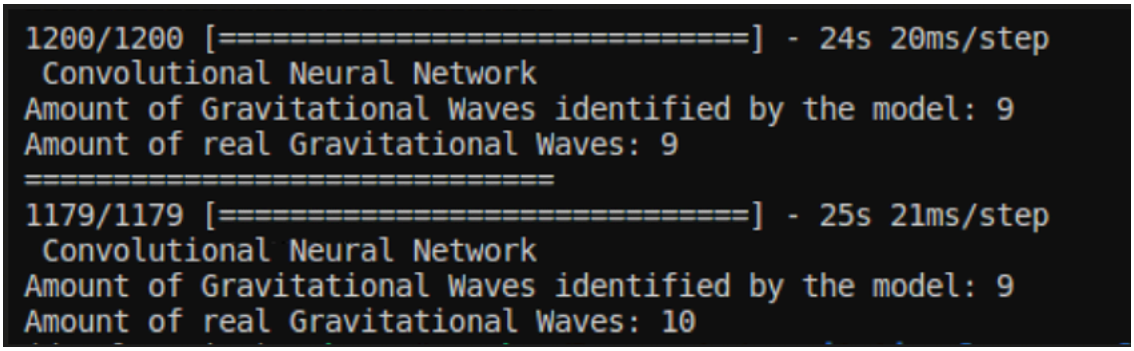
We trained this model and then validated it. We used the standard way of training suggested by the library and then we used cross-validation[45] over five epochs. The validation set was the very same one labeled and suggested "as is" in the gravity spy paper; this was intentional due to the fact that it will allow us to contrast findings and classification results over the former results.

Figure 8.2.1 - The 1st model predict against the "validation" set



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 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, Figure 8.1.1.3 shows the confusion matrix obtained in the gravity spy classification by a CNN. In this matrix we see that the classification performance of a “chirps” is 1.0, as well. Which is encouraging and gives us a cue about the nature of the current set. These findings could be reproduced executing the scripts found in the code base[46].

Figure 8.2.2 - The model classification of the validation and test set got the right images.

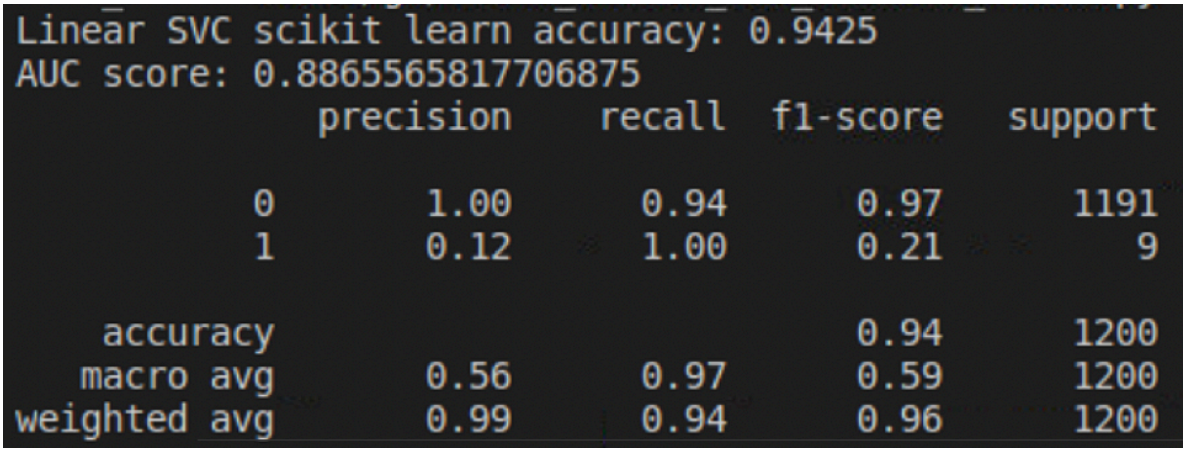


To understand this performance and the rationale behind it, we performed several other trainings and classifications. For our 2nd model we changed the following:

- 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 agains the labeled “test” set.

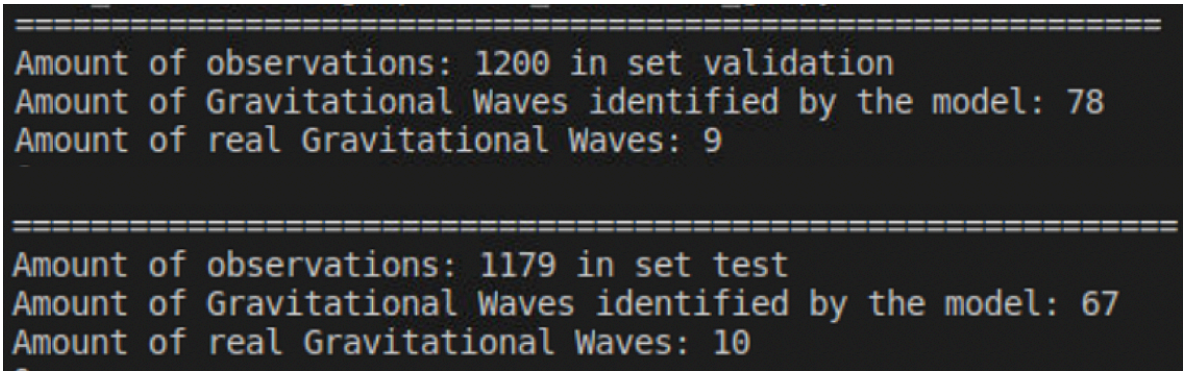
The results we found where the same as long as the C parameter was over 0.000075; as far as we approximated to this value or surpassed it the entire model precision, recall, ROC/AUC and f1 parameter started to deplete.

Figure 8.2.3 - Once the C parameter approximated to 0.000075 the metrics fell.



Therefore, the model prediction performance in the original "validation" and "test" set were as the ones showed in figure 8.2.4.

Figure 8.2.4 - The prediction's performance turned out to "predict" much more false-positives after we diminished the C parameter.



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[47]). The metrics confirmed were below 0.35 approximately.

- We tried to use our very own training and test sets with similar results across all the training experiments previously tested.
- We tried with different values of C: 0.5, 0.1, these maintanined the ROC/AUC, precission, 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[48].

The predict's scripts and their metrics can be found from the code base[49].

The training times for the optimal configuration, were as:

- Train LINEAR SVC --- 75.95 seconds
- Validation LINEAR SVC --- 2.68 seconds
- Cross Validation LINEAR SVC --- 59.98 seconds

The prediction times for the trained model were:

- Predict LINEAR SVC, VALIDATION — 3.0529794692993164 seconds —
- Predict LINEAR SVC, TEST --- 3.1237270832061768 seconds —

We conclude that the spectrograms constitute a very regular set of images. These are bicolor, very cantered and regular. Given those, and considering the small magnitudes at which the hyperplane drawn by the algorithm was starting to falter (below 0.0005). This gives us the idea that the support vectors between every 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, for example).

8.3 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, everyone constituent of these O1 and O2 measurements. We configured the network with different layers; which were mere adaptations of an exercise from the Deep Learning book[50] example for classification of digits of the MNIST classic dataset:

1. Conv2D(32, (3,3), activate='relu', input_shape=(140,170,1))
2. MaxPooling((2,2))
3. Conv2D(64, (3,3), activate='relu')
4. MaxPooling((2,2))
5. Conv2D(64, (3,3), activate='relu')
6. Flatten()
7. Dense(64, activate='relu')
8. Dense(2, activate='softmax')

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" spectrograms; which were below the 0.008 of the total. We trained over 50 epochs but the network fail to identify one of the images in the validation or testing sets.

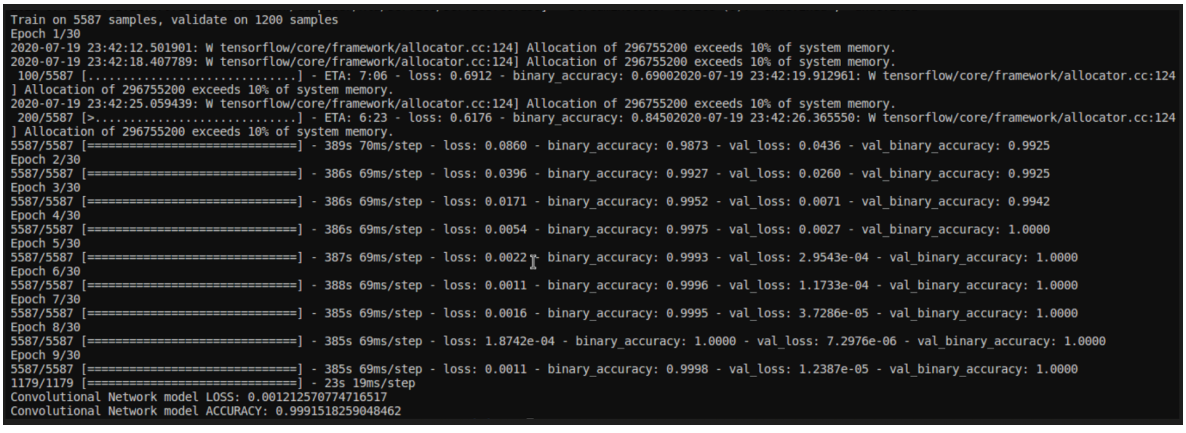
We found that the network "deepness" was the problem. With the given positives' set (41 in 5587), we needed to build up a bigger network. Therefore, we changed the model for a similar one

found in the same book[51] 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:

1. Conv2D(32,(3,3), activation="relu", input_shape=(140,170,1))
2. MaxPooling2D(2,2)
3. Conv2D(64,(3,3), activation="relu")
4. MaxPooling2D(2,2)
5. Conv2D(128,(3,3), activation="relu")
6. MaxPooling2D(2,2)
7. Conv2D(128,(3,3), activation="relu")
8. MaxPooling2D(2,2)
9. Flatten()
10. Dense(512, activation='relu')
11. Dense(1, activation=‘sigmoid’)

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

Figure 8.3.1 - Generated during the training session for the Convolution Neural Network approach.



As we can see in figure 8.3.1, we intended to train the network for 30 epochs, with a validation set and a clause that stopped the training as soon as the network ceased to show any improvements. The trained found that the loss and accuracy ceased to improve after the ninth epoch. Therefore, we tested the model against the "testing set", which thought the following results:

- Convolutional Network model LOSS: 0.00121 (approximately)
- Convolutional Network model ACCURACY: 0.99915 (approximately)

The history for these epochs were as follows:

Figure 8.3.2 - ConvNet accuracy vs validation set

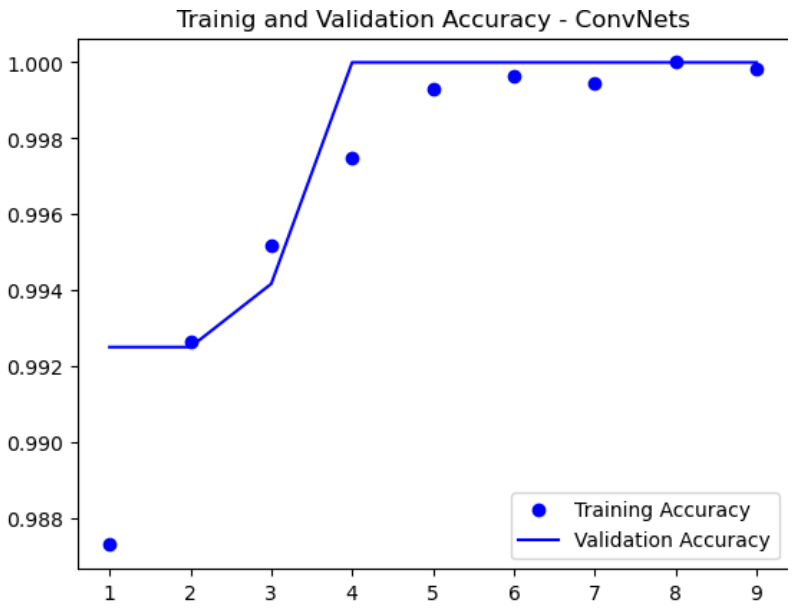
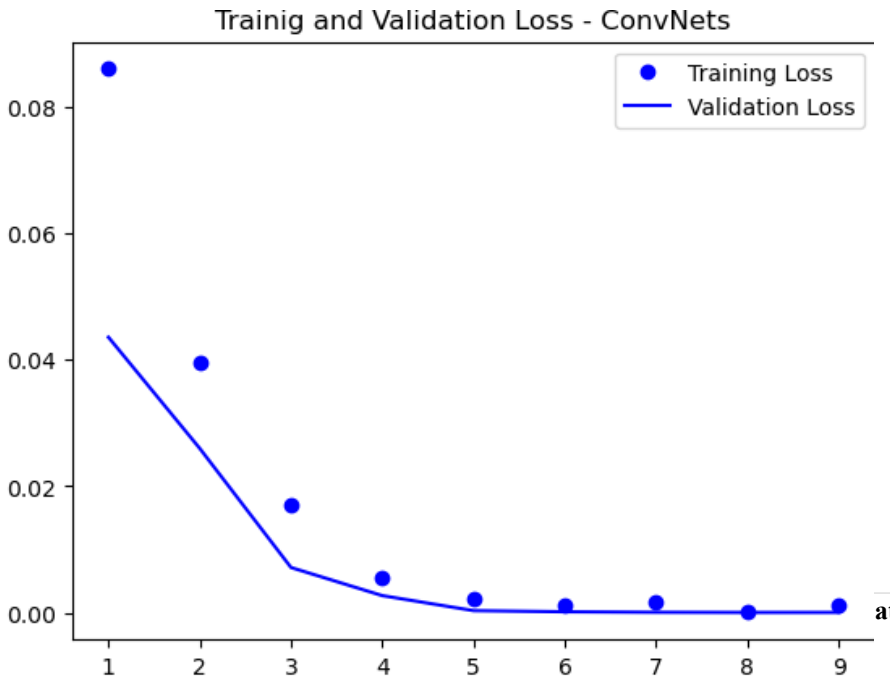
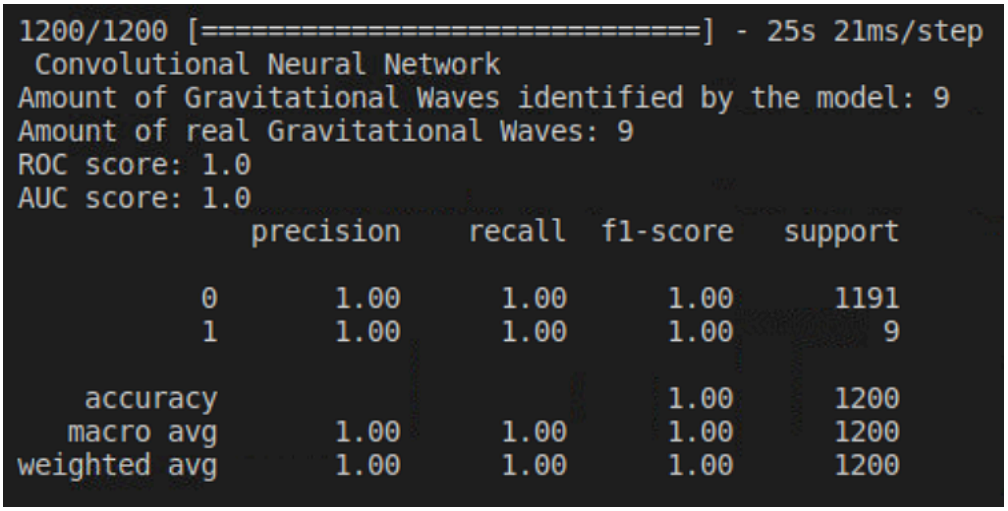


Figure 8.3.3 - ConvNet loss vs validation set



As can be appreciated in both images (Figure 8.3.2 and 8.3.3), the training and validation sets converges during cross-validation training for almost all nine epochs; which ensures that we are avoiding overfitting issues.

Figure 8.3.4 - ConvNet precision, recall, f1 and ROC/AUC for the “validation” set.

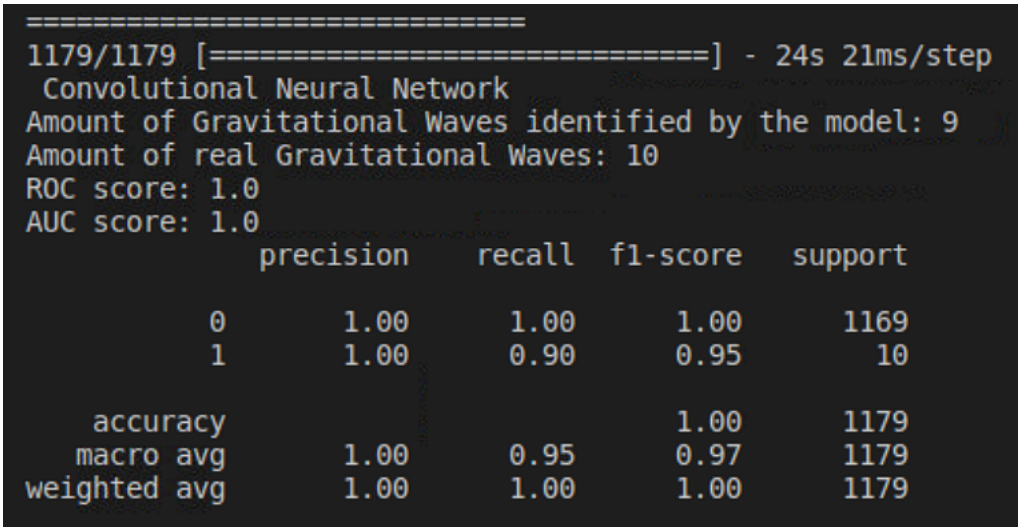


```
1200/1200 [==============================] - 25s 21ms/step
 Convolutional Neural Network
Amount of Gravitational Waves identified by the model: 9
Amount of real Gravitational Waves: 9
ROC score: 1.0
AUC score: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1191
           1       1.00      1.00      1.00         9

    accuracy                           1.00      1200
   macro avg       1.00      1.00      1.00      1200
weighted avg       1.00      1.00      1.00      1200
```

Figure 8.3.5 - ConvNet precision, recall, f1 and ROC/AUC for the “test” set.



```
==============================
1179/1179 [==============================] - 24s 21ms/step
 Convolutional Neural Network
Amount of Gravitational Waves identified by the model: 9
Amount of real Gravitational Waves: 10
ROC score: 1.0
AUC score: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1169
           1       1.00      0.90      0.95        10

    accuracy                           1.00      1179
   macro avg       1.00      0.95      0.97      1179
weighted avg       1.00      1.00      1.00      1179
```

The current convolutional neural network reached an asymptotic curve around epoch 5; we saw afterwards that the network classified all the validation sets gravitational waves (9 out of 9) and all the same images from the tests set (10 out of 10). Given that the aim of this work was to classify "at least one image", we think that this model is appropriately trained for its scope.

The convolutional network training happens in:

Train CONVNET --- 1206.84 seconds (20 minutes) —

The convolutional network prediction happens in:

Predict CONVNET --- 24.21 to 25.21 seconds ---

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.

8.4 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 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 are going to try to classify the "Chirps" (gravitational waves) as in the other methods. The configuration will entail the usage of a LSTM (Long-Short Term Memory) configuration[52]. We used the following configuration in a "sequential" model:

- LSTM(4, return_sequences=True)(input)
- LSTM(4, return_sequences=True)
- LSTM(4)
- Dense(1, activation='sigmoid')(x)

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)
- loss='binary_crossentropy'
- 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 69 epoch, we reached the peaks of improvement for the model. With the results showed in table 8.4.1. 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 0.75555 for the class 1 (the "Chirp").

Table 8.4.1 - The RecNet, first classification model.

Using TensorFlow backend.

Model: "model_1"

_____

Layer (type) Output Shape Param #

==========

input_1 (InputLayer) (None, 140, 170) 0

_____

lstm_1 (LSTM) (None, 140, 4) 2800

_____

lstm_2 (LSTM) (None, 140, 4) 144

_____

lstm_3 (LSTM) (None, 4) 144

_____

dense_1 (Dense) (None, 1) 5

==========

Total params: 3,093

Trainable params: 3,093

Non-trainable params: 0

Train on 5587 samples, validate on 1200 samples

Epoch 69/90

5587/5587 [==========] - 6s 1ms/step - loss: 0.1810 - binary_accuracy: 0.9962 - val_loss: 0.0969 - val_binary_accuracy: 0.9967

Train REVNET --- 424.92 seconds (7 minutes)—

1179/1179 [==========] - 1s 714us/step

Recurrent Network model LOSS: 0.1025800785338909

Recurrent Network model ACCURACY: 0.995759129524231

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

Table 8.4.2 - The trained LSTM RecNet performance results.

Validation set

1200/1200 [==============================] - 1s 849us/step

Predict RECNET --- 1.020552158355713 seconds ---

Recursive Neural Network

Amount of Gravitational Waves identified by the model: 10

Amount of real Gravitational Waves: 9

ROC score: 0.9991603694374475

AUC score: 0.9991603694374475

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1191
1	0.90	1.00	0.95	9
accuracy			1.00	1200
macro avg	0.95	1.00	0.97	1200
weighted avg	1.00	1.00	1.00	1200

Test Set

1179/1179 [==============================] - 1s 435us/step

Predict RECNET --- 0.513575553894043 seconds ---

Recursive Neural Network

Amount of Gravitational Waves identified by the model: 10

Amount of real Gravitational Waves: 10

ROC score: 0.8721984602224123

AUC score: 0.8721984602224123

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1169
1	0.80	0.80	0.80	10
accuracy			1.00	1179
macro avg	0.90	0.90	0.90	1179
weighted avg	1.00	1.00	1.00	1179

Figure 8.4.3 - The trained LSTM Loss

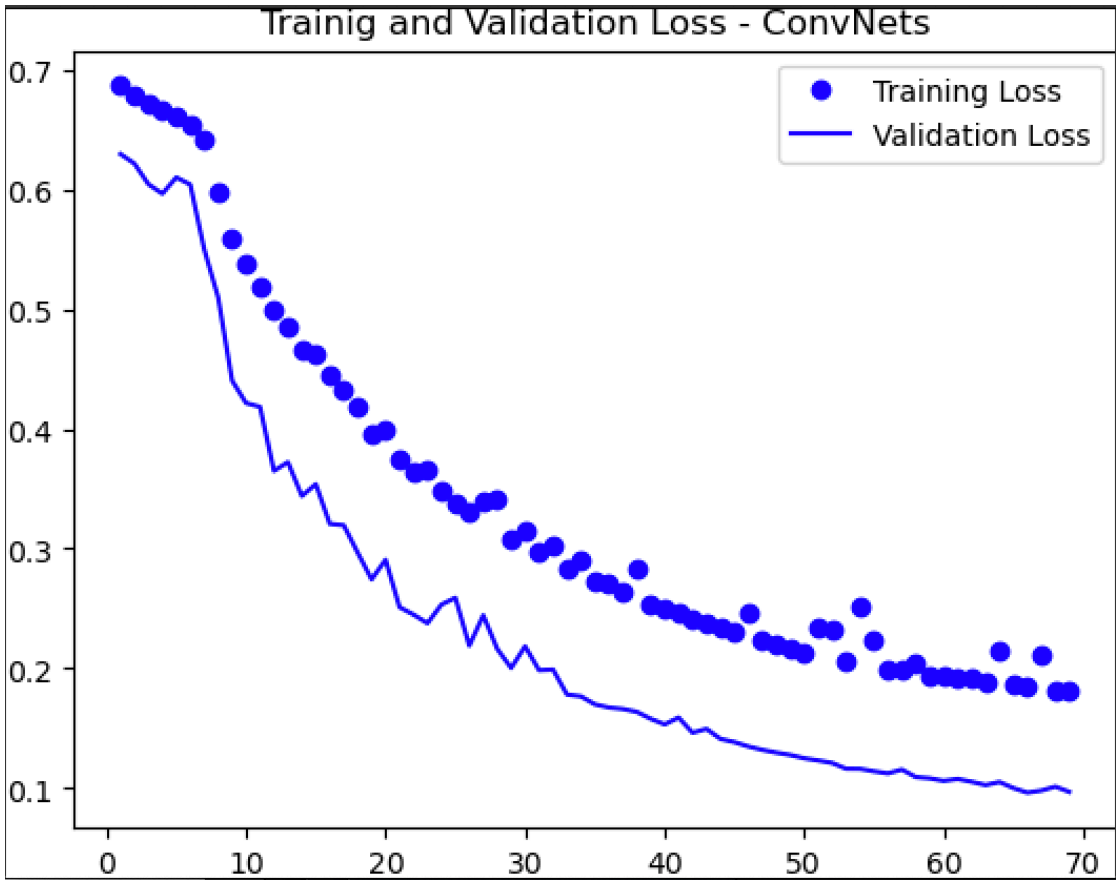
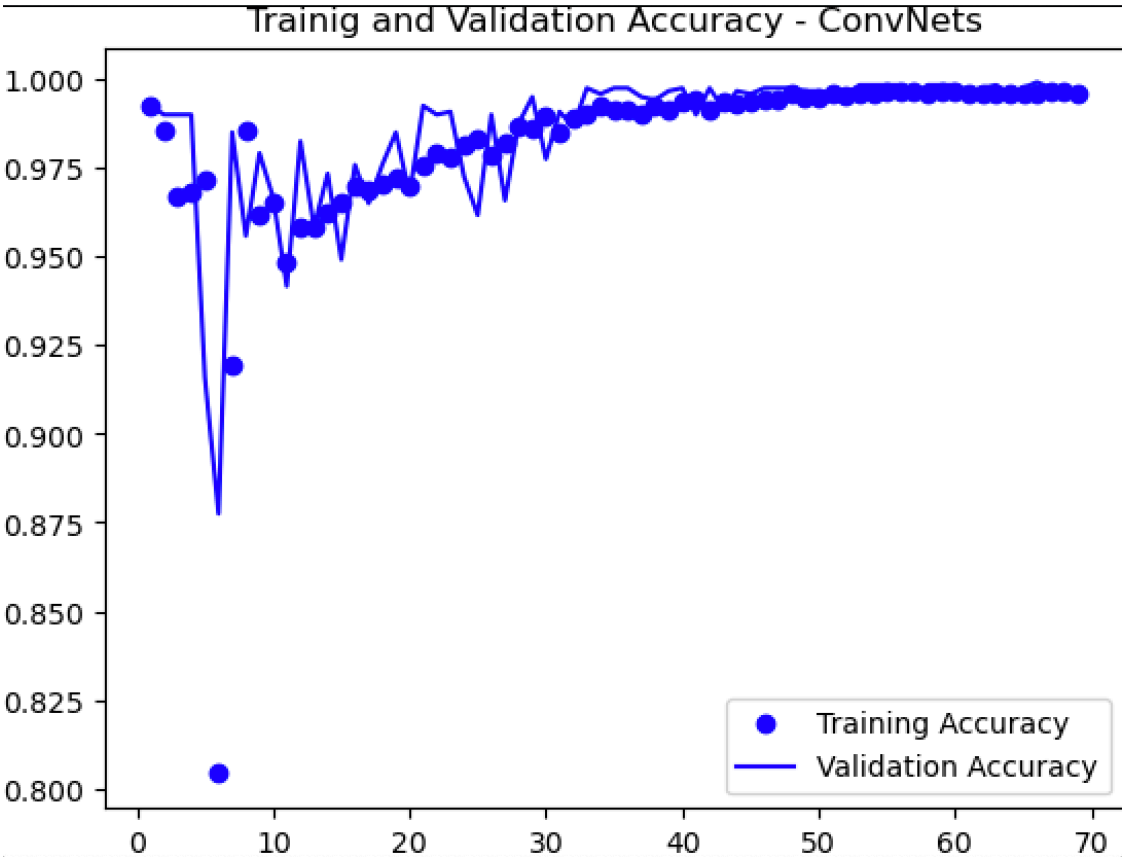


Figure 8.4.4 - The trained LSTM Accuracy



As we can observe, 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 8.4.2). The model classified the majority of the gravitational waves, introducing some false-positives. Besides, the loss and accuracy graphs for each epoch shows that the model's overfitting risk, for the given set, is being avoided.

8.5 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 model configuration asserted was the default for the lightgbm[53] 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 8.5.1 - The light gbm default model training results

Train Light GBM --- 276.39 seconds ---

Validation Light GBM --- 0.16 seconds ---

Light GBM scikit learn basic score: 0.9958

Cross Validation Light GBM --- 746.83 seconds ---

Light GBM scikit learn cross-val score: 0.9958

ROC score: 0.9991603694374476

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1191
1	0.75	0.67	0.71	9
accuracy			1.00	1200
macro avg	0.87	0.83	0.85	1200
weighted avg	1.00	1.00	1.00	1200

Table 8.5.2 - The light GBM default model predict results

Validation Set

==========

Predict Light GBM --- 0.17 seconds ---

Amount of observations: 1200 in set "validation"

Amount of Gravitational Waves identified by the model: 8

Amount of real Gravitational Waves: 9

ROC score: 0.9991603694374476

Now the TEST set...

Test Set

==========

Predict Light GBM --- 0.25 seconds ---

Amount of observations: 1179 in set "test"

Amount of Gravitational Waves identified by the model: 9

Amount of real Gravitational Waves: 10

ROC score: 0.9998289136013687

As we see in table 8.5.2, the model is capable of classifying almost all the gravitational waves found in the dataset. However, after analysis, 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 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.

Table 8.5.3 - The light GBM balanced model training results

Train Light GBM --- 253.49 seconds ---

Validation Light GBM --- 0.15 seconds ---

Light GBM scikit learn basic score: 0.9992

Cross Validation Light GBM --- 912.8812453746796 seconds ---

Light GBM scikit learn cross-val score: 0.9992

ROC score: 0.9860994495755201

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1191
1	1.00	0.89	0.94	9
accuracy			1.00	1200
macro avg	1.00	0.94	0.97	1200
weighted avg	1.00	1.00	1.00	1200

Table 8.5.4 - The light GBM balanced model predict results

Validation Set

==========

Predict Light GBM --- 0.13507890701293945 seconds ---

Amount of observations: 1200 in set validation

Amount of Gravitational Waves identified by the model: 8

Amount of real Gravitational Waves: 9

ROC score: 0.9860994495755201

Now the TEST set…

==========

Predict Light GBM --- 0.12882208824157715 seconds ---

Amount of observations: 1179 in set test

Amount of Gravitational Waves identified by the model: 9

Amount of real Gravitational Waves: 10

ROC score: 0.9998289136013687

As we see in table 8.5.3 some metrics improved against the “default model”; but after analysing the classified images we confirmed that the classification missed one signal in every case, but didn’t produced any false positives. We tried with other two models that balanced the two classes.

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 8.5.5 - The light GBM manually balanced weights model training results

Train Light GBM --- 4.035710096359253 seconds ---

Validation Light GBM --- 0.1647171974182129 seconds ---

Light GBM scikit learn basic score: 0.9075

Cross Validation Light GBM --- 945.2859272956848 seconds ---

Light GBM scikit learn cross-val score: 0.9075

ROC score: 0.9742513294150573

	precision	recall	f1-score	support
0	1.00	0.91	0.95	1191
1	0.07	0.89	0.13	9
accuracy			0.91	1200
macro avg	0.53	0.90	0.54	1200
weighted avg	0.99	0.91	0.94	1200

Table 8.5.6 - The light GBM balanced model predict results

Validation Set

==========

Predict Light GBM --- 0.14285850524902344 seconds ---

Amount of observations: 1200 in set validation

Amount of Gravitational Waves identified by the model: 118

Amount of real Gravitational Waves: 9

ROC score: 0.9742513294150573

Now the TEST set...

==========

Predict Light GBM --- 0.19250726699829102 seconds ---

Amount of observations: 1179 in set test

Amount of Gravitational Waves identified by the model: 135

Amount of real Gravitational Waves: 10

ROC score: 0.9879384088964926

In spite the fact that most metrics seems to fare pretty high (above 0.90) for most except for the f1 and the images identifies are conclusive: this model increases the amount of false positives identified significantly.

However, we testes 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.

Table 8.5.7 - The light GBM manually exploded balanced weights model training results

Train Light GBM --- 407.55 seconds ---

Validation Light GBM --- 0.13 seconds ---

Light GBM scikit learn basic score: 0.9975

Cross Validation Light GBM --- 916.49 seconds ---

Light GBM scikit learn cross-val score: 0.9975

ROC score: 0.9475697359828341

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1191
1	0.80	0.89	0.84	9
accuracy			1.00	1200
macro avg	0.90	0.94	0.92	1200
weighted avg	1.00	1.00	1.00	1200

Table 8.5.8 - The light GBM exploded balanced model predict results

Validation Set

==========

Predict Light GBM --- 0.1667797565460205 seconds ---

Amount of observations: 1200 in set validation

Amount of Gravitational Waves identified by the model: 10

Amount of real Gravitational Waves: 9

ROC score: 0.9475697359828341

Now the TEST set…

==========

Predict Light GBM --- 0.18632841110229492 seconds ---

Amount of observations: 1179 in set test

Amount of Gravitational Waves identified by the model: 10

Amount of real Gravitational Waves: 10

ROC score: 0.9999999999999999

As we may observe in table 8.5.7 and 8.5.8 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 classify ing most of the images in the validation and test dataset right after balancing the amount of data searched.

9. Conclusions

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 was more difficult to configure, and we saw algorithms that where supposed to fare pretty well with structured data and performing well still with this dataset. As we explained in **appendix A**, this is a pretty particular, very well categorised dataset. Most of the images are in the center, 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.

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 an model that has overfitted the dataset.

- **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 8.1.1.3, 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 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 nature of the dataset.

- **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, looking for classification of most of the know "glitches" we can find in a spectrogram analysed by a physicist. These models could be trained to classify in a binary fashion each single kind of glitch. Therefore, opening the door to a different approach: classifying specific events instead of all 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.

This work, 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.

10. 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[54]. 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[55]. 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*[56] compatible format. In this format, which entails the usage of the function *imshow*[57],

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[58].

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)

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] 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

[25] GWOSC(2016), "The O2 database release" [Dataset and coding libraries], recovered from https://www.gw-openscience.org/O2/

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

[27] The pandas development team (Version 1.0.3), recovered from https://pandas.pydata.org/docs/

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

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

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

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

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

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

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

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

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

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

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

[39] "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

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

[41] "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

[42]"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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[58]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