G2Net

Gravitational Wave Detection

Find gravitational wave signals

from binary black hole collisions

Kaggle competition

Table of Content

[Introduction 4](#_Toc77092575)

[Story 4](#_Toc77092576)

[Objective 4](#_Toc77092577)

[Inventory 5](#_Toc77092578)

[Kaggle 5](#_Toc77092579)

[Local 5](#_Toc77092580)

[Framework 5](#_Toc77092581)

[Data 5](#_Toc77092582)

[Competition 5](#_Toc77092583)

[Workplan 5](#_Toc77092584)

[Notebooks 5](#_Toc77092585)

[Starter from community 6](#_Toc77092586)

[Dataset starter 6](#_Toc77092587)

[Inference 6](#_Toc77092588)

[Data handling 6](#_Toc77092589)

[Limitation 6](#_Toc77092590)

[Conclusion 7](#_Toc77092591)

[Simple EDA 7](#_Toc77092592)

[Data range 7](#_Toc77092593)

[Wave plot 8](#_Toc77092594)

[Spectrogram 8](#_Toc77092595)

[Quote from Kaggle: 8](#_Toc77092596)

[QCT 9](#_Toc77092597)

[Improvement 10](#_Toc77092598)

[Tensorflow Dataset 10](#_Toc77092599)

[Data ingestion 11](#_Toc77092600)

[Modeling 11](#_Toc77092601)

[Submission 11](#_Toc77092602)

[Conclusions 11](#_Toc77092603)

# Introduction

## Story

Inspired from:

<https://en.wikipedia.org/wiki/First_observation_of_gravitational_waves>

In 2015 September the 14th, the first direct observation of gravitational waves happened and officially confirmed the 11th of February 2016.

This event had been recorded by the two Laser Interferometer Gravitational-Wave Observatory (LIGO) located in the United States of America (Hanford and Livingston).

A third Large Interferometer named Virgo located in Europe was upgraded shortly after improving the Gravitational Waves potentialities.

Since then, these 3 Larges Interferometers are actively searching for detectable Gravitational Waves (GW).

In short, the interferometers are built around an L structure where laser emits photon along the few kilometer’s arms, these photons are reflected at the extremities, then is measured their travel time.

When a strong enough event happens the GW impacts photons travel time (rather curve the space structure which alters the travel length, hence modify the travel time).

By strong event here we are talking of black hole collision or at least neural star collision. Yes, this is a scary universe…

## Objective

Long story short, the objective is predicting whether 3 channels recorded events match gravitational wave. A simple yes or no binary classification problem.

Quoting Kaggle:

<https://www.kaggle.com/c/g2net-gravitational-wave-detection/data>

In this competition you are provided with a training set of time series data containing simulated gravitational wave measurements from a network of 3 gravitational wave interferometers (LIGO Hanford, LIGO Livingston, and Virgo). Each time series contains either detector noise or detector noise plus a simulated gravitational wave signal. The task is to identify when a signal is present in the data (target=1).

The parameters that determine the exact form of a binary black hole waveform are the masses, sky location, distance, black hole spins, binary orientation angle, gravitational wave polarization, time of arrival, and phase at coalescence (merger). These parameters (15 in total) have been randomized according to astrophysically motivated prior distributions and used to generate the simulated signals present in the data, but are not provided as part of the competition data.

## Inventory

### Kaggle

Kaggle provide with cloud (remote) Jupyter notebooks, 16GB RAM, multi-processor, 20GB local space.

Kaggle notebooks can be GPU enhanced but limited to 35 hours per week, counter is running for the notebooks not the GPU usage time; therefore, to be activated cautiously, that’s why the below local environment will be leveraged for most of the trainings.

Also note when running in GPU mode, less CPU are available so care with heavy CPU preprocessing, in fact we’ll rely on intermediate preprocessed data.

### Local

A local computer based on CPU Intel 4790K, 16GB RAM, GPU RTX2060, and SSD.

### Framework

Jupyter python, with TensorFlow Framework chosen as per my competencies.

Leveraging at most the TensorFlow API including Datasets, addons, and integrated Keras is a personal bet I hope will be rewarding in the future, but may be tricky at some point due to the Kaggle still in 2.4.0 Tensorflow update level.

### Data

There are 560 000 numpy records available for training.

Each data sample (npy file) contains 3 time series (1 for each detector), each spanning for 2 sec, and sampled at 2,048 Hz. Something like 96KB data per sample.

That’s around 60GB to deal with when including test data.

### Competition

Root of the competition is at <https://www.kaggle.com/c/g2net-gravitational-wave-detection>

Started the 30th of June 2021

Runs for 3 months

# Workplan

First step is identifying a starter notebook will help getting hands on Kaggle environment and uses.

the second step which is the first focus of this study is rewriting the data ingestion and transformation mechanisms matching at best Tensorflow build in Dataset mechanism.

Last in line the modeling will leverage previously generated data Tensorflow Dataset way trying to predict whether each record matches a Gravitational Wave as expected by the competition.

## Notebooks

Based on an existing starter notebook the plan is building two notebooks, the first focusing on data ingestion and transformation, the second handling the machine learning (ML) model based on the transformation schema defined in the first.

## Starter from community

« TF G2Net EDA and Starter » <https://www.kaggle.com/mrigendraagrawal/tf-g2net-eda-and-starter>

The mentioned provides with a functional notebook for data ingestion, transformation, and basic model run leveraging transfer learning from an EfficientNet B0 model.

What bothered me at first was the Dataset mechanism relying on “Sequence” imported from “keras.utils” which as per my understanding has now most of its complexity integrated now days Tensorflow Dataset API.

In line with the “Sequence”, the notebooks mix Tensorflow and former Keras functions. This may be source of confusion and as per my short experience bug friendly. Hence my will to rewrite this portion leveraging at best Tensorflow API.

Yet don’t get confused with my concerns regarding this initial starter notebook, it is a good job fully functional and definitely a good starting point permitting training and submission out of the box.

## Dataset starter

The first notebook proposed will propose rewritten data ingestion leveraging Tensorflow Dataset API.

Kaggle notebook available at <https://www.kaggle.com/vincentdumetz/g2net-tf-dataset-starter>

## Inference

The second notebook will handle inference based on the Dataset workflow defined in previous notebook.

Kaggle notebook available at <https://www.kaggle.com/vincentdumetz/g2net-tf-dataset-inference>

# Data handling

Let’s have a look at the data and their environment.

Below some considerations on the limits

### Limitation

<https://www.tat.physik.uni-tuebingen.de/~kokkotas/Teaching/NS.BH.GW_files/GW_Physics.pdf>

For a gravitational wave with amplitude h ∼ 10−21 and detector arm- length 4 km (such as LIGO), this will induce a change in the arm-length of about ∆L ∼ 10−16.

#### Photon shot noise.

This limitation in the detector’s sensitivity due to the photon counting uncertainty is known as photon shot noise. For a typical laser interferometer, the **photon shot noise is the dominant source of noise for frequencies above 200 Hz**, while its power spectral density Sn(f) for frequencies 100-200Hz is of the order of ∼ 3 × 10−23 √Hz.

#### Radiation pressure noise

As we have seen, the photon shot noise decreases as the laser power increases, while the inverse is true for the noise due to radiation pressure fluctuations. If we try to minimize these two types of noise with respect to the laser power, we get a minimum detectable strain for the optimal power via the very simple relation (3.10) which for the LIGO detector (where the mass of the mirrors is ∼100 kg and the arm length is 4 km), **for observation time of 1ms, gives hmin ≈ 10−23**.

#### Quantum limit

An additional source of uncertainty in the measurements is set by Heisenberg’s principle, which says that the knowledge of the position and the momentum of a body is restricted from the relation ∆x·∆p ≥ h.

Surprisingly, this is identical to the optimal limit that we calculated earlier for the other two types of noise.

#### Seismic noise

At frequencies below 60 Hz, the noise in the interferometers is dominated by seismic noise. This effect is strongly suppressed by properly designed suspension systems. **Still, seismic noise is very difficult to eliminate at frequencies below 5-10 Hz.**

#### Residual gas-phase noise

The statistical fluctuations of the residual gas density induce a fluctuation of the refraction index and consequently of the monitored phase shift. For this reason, the laser beams are enclosed in pipes over their entire length. Inside the pipes a high vacuum of the order of 10−9 Torr guarantees elimination of this type of noise.

### Conclusion

From the upper consideration we should focus on wave length from 15 to 200.

## Simple EDA

Remember records run over 2 seconds sampled at 2048Hz.

### Data range

Below are some stats worth mentioning over thousands of records

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Site | Mean Std | Min | max | Mean Mean |
| Site 0 (LIGO 1) | 7.269105e-21 | -2.883983e-20 | 3.534220e-20 |  |
| Site 1 (LIGO 2) | 7.315387e-21 | -2.918349e-20 | 2.801213e-20 |  |
| Site 2 (Virgo) | 1.805447e-21 | -8.079410e-21 | 7.811783e-21 |  |

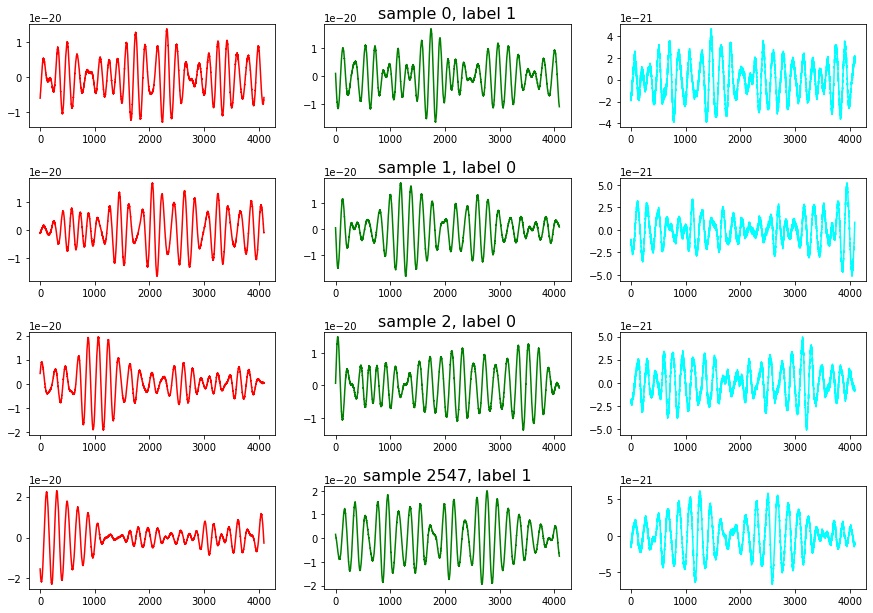
Stats are consistent between the two LIGOs but we see Virgo has different range with less amplitude and standard deviation.

Before processing data will be divided by their maximum scaling to 1 as max per channel in each recorded event.

Yet we may wonder about the impact of the Virgo being an ‘other type’ of Large Interferometer. This may deserve a special attention, yet will not be considered here further.

### Wave plot

Below are a few samples plotted, label 1 means a GW is present while label 0 means no GW.



## Spectrogram

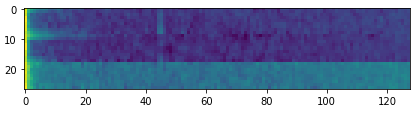
### Quote from Kaggle:

“The integrated signal-to noise ratio (SNR) is classically the most informative measure of how detectable a signal is and a typical level of detectability is when this integrated SNR exceeds ~8. This shouldn't confused with the instantaneous SNR - the factor by which the signal rises above the noise - and in nearly all cases the (unlike the first gravitational wave detection GW150914) these signals are not visible by eye in the time series.”

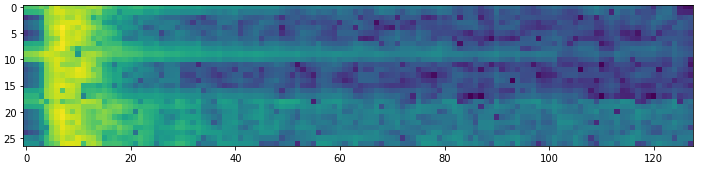
The first approach proposed in the starter was leveraging the “librosa” library transforming the data into “melspec” spectrograms.

First code did leverage the “melspectrogram” function from “librosa” I did alter a bit on “fmin” an “fmax” conforming to previous observation.

Below a example of spectrogram



With fmin and fmax adapted:



melspecs = []

for j in range(3):

melspec = librosa.feature.melspectrogram(waves[j] / max(waves[j]),

sr=4096, n\_mels=128, fmin=10, fmax=200)

melspec = librosa.power\_to\_db(melspec)

melspec = melspec.transpose((1, 0))

melspecs.append(melspec)

image = np.vstack(melspecs)

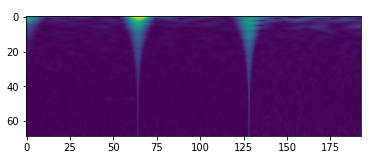
return image

By the way one should pay attention to the 3 waves being independently transformed then stacked generating the final image. The result is grey scaled and coloring mapped to intensity.

Yet the Transfer Learning did perform well leveraging an Efficientnet B0 achieving over .80 Area Under the Curve which is the binary metric chosen by Kaggle for this competition.

## QCT

But few days after the owner of the starter updated with an improved signal transformation, the Constant-Q Transform (CQT) here integrated via the nn.audio Torch library in the “CQT1992v2” specific implementation.

The output graphical output is horizontally stacked and the pikes match the jointures.:

Initial code being:

def increase\_dimension(idx,is\_train,transform=CQT1992v2(sr=2048, fmin=20, fmax=1024, hop\_length=64)): # in order to use efficientnet we need 3 dimension images

waves = np.load(id2path(idx,is\_train))

waves = np.hstack(waves)

waves = waves / np.max(waves)

waves = torch.from\_numpy(waves).float()

image = transform(waves)

image = np.array(image)

image = np.transpose(image,(1,2,0))

return image

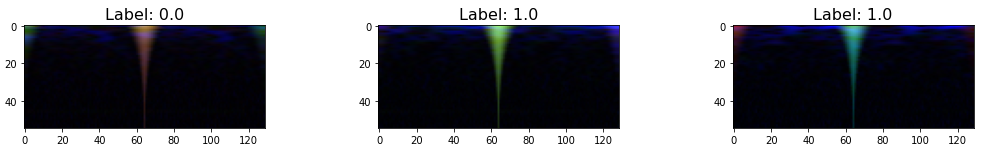
## Improvement

Until now the transformations remain grey scaled and the starter’s owner relies on a convolutional 2D trick in order to feed the 3 RGB channels expected by the EfficientNet for modeling.

My little idea is rather than stacking the transformations let’s populate each of the RBG channels with one specific instrument.

Yet the visible graphical part which seems of interest as seen before with the jointure is on the extremities of the graphical output, therefore and until better mastering of these transformation techniques, I concatenate each wave with itself achieving the centering of the visual.

The output for 3 different samples becomes:



Each color matches an instrument: Red for LIGO1, Green for LIGO2, and Blue for Virgo. Yet don’t be fooled by the coloring: They are far from conclusive detecting a Gravitational Wave.

At this stage I definitely need to improve my skills in signal analysis and transformation, but for now focusing on the modeling fixing technical and physical limitation I’m facing:

* The transformations are CPU killer,
* Relying on intermediate pre-processed dataset I can @home but the 20GB output Kaggle limitation can’t handle the initial 60GB data transformation
* Therefore, I can hardly leverage the GPU as Kaggle diminish to 2 CPU cores when activating GPU… To be continued.
* Last but not least the CQT1992v2 is from Torch library which impose relying on external py\_function while inside the Tensorflow pipeline.

# Tensorflow Dataset

Let’s look at the Tensorflow Dataset implementation now

The main challenges are:

* Tensorflow version
  + Kaggle Tensorflow is version 2.4.1 which is quite old now and Dataset handling has been greatly improved in 2.5 (not to mention nightly tf is in 2.7)
  + Activating the GPU after upgrading Tensorflow to 2.5 requires specific CUDA libraries handling not to mention the start overhead due to tf upgrade not mentioning the packages conflict with Torch version. Long story short: I don’t have time for this road right now.
* Tensorflow Dataset do not handle the data former numpy format naturally
  + This I fixed!

That’s it let’s start with data Ingestion

## Data ingestion

As stated, the numpy format is not handled but the below function gives us the header size:

# Modeling

# Submission

# Conclusions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| function | alpha | Validation accuracy | Batch to train | Time to train | Batch time (m:ss) |
| Leaky ReLU (Baseline) | .3 | **.5608** | 32 | 2h59 | **5:59** |
| ReLU | N/A | .4271 | 25 | 2h29 | 5:96 |
| PReLU | N/A | .4192 | 27 | 3h49 | 8:48 |
| ELU | .25 | .5342 | 32 | 4h02 | 7:56 |
| GELU | N/A | .4896 | 32 | 4h24 | 8:25 |
| SELU | N/A | .4567 | 32 | 3h07 | 5:84 |
| SWISH | N/A | .4671 | 32 | 3h27 | 6:46 |