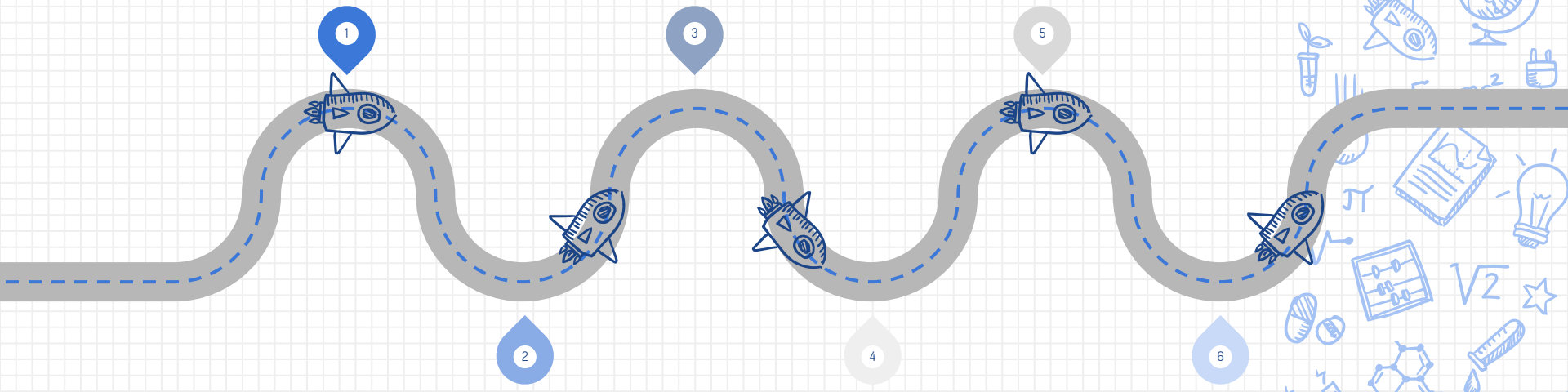
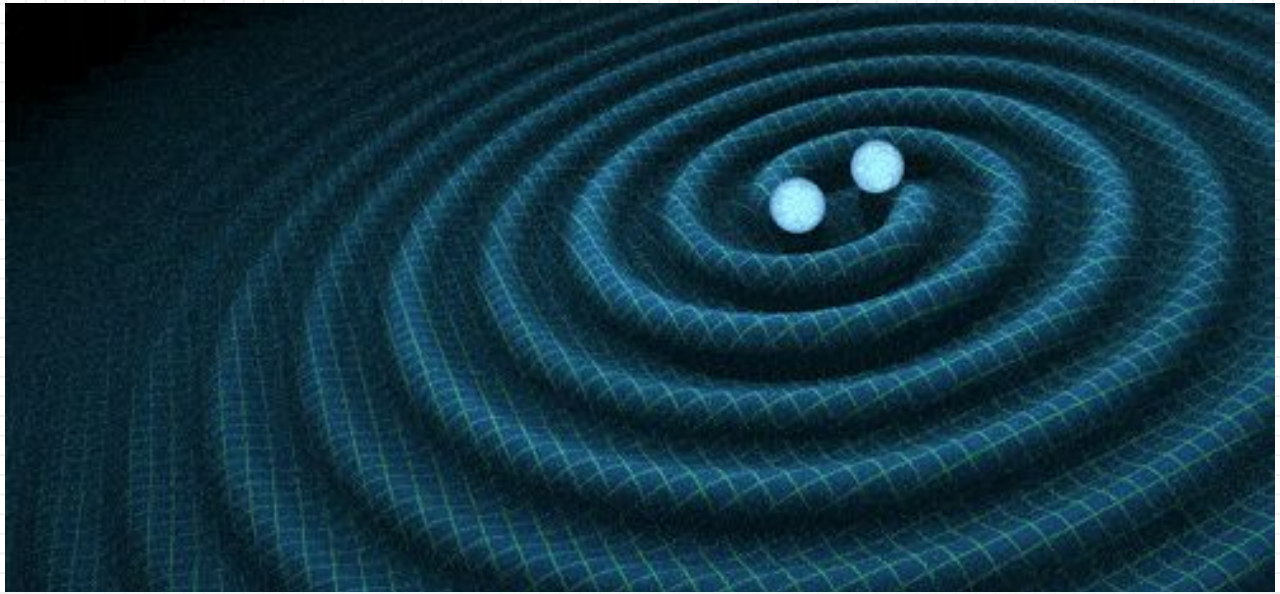
Gravitational Waves
Do You See it?

Roadmap


1
What are Gravitational Waves?
2
Why are GW Important?
3
How are GW Detected?
4
How can Machine Learning help?
5
Modelling and Evaluation
6
Conclusion

What are Gravitational Waves?

A picture is worth a thousand words



"Ripples" in spacetime caused by accelerating massive celestial bodies

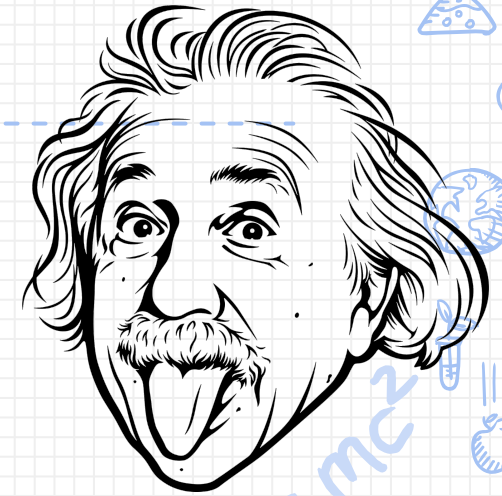
Energy radiated as gravitational waves (think dropping pebble in pond)

Source: https://www.sbs.com.au/topics/science/space/article/2016/02/12/gravitational-waves-detected-scientists-historical-first

Why are GW Important?

Source: https://www.nature.com/articles/d41586-018-04157-6

Why are GW Important?



- ✘ One of the final pieces in Einstein's General Theory of Relativity
- ✘ Opens new "windows" to study the Universe in addition to EM radiation

Ok, mainly to answer

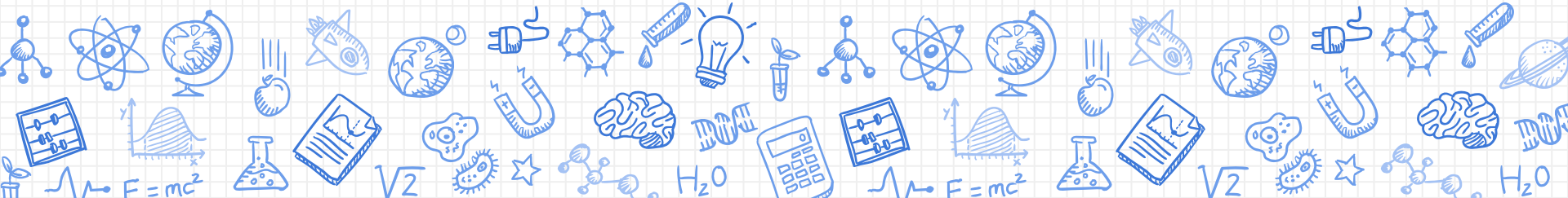
- ✘ How old is the Universe?
- ✘ How big is the Universe?

Problem Statement

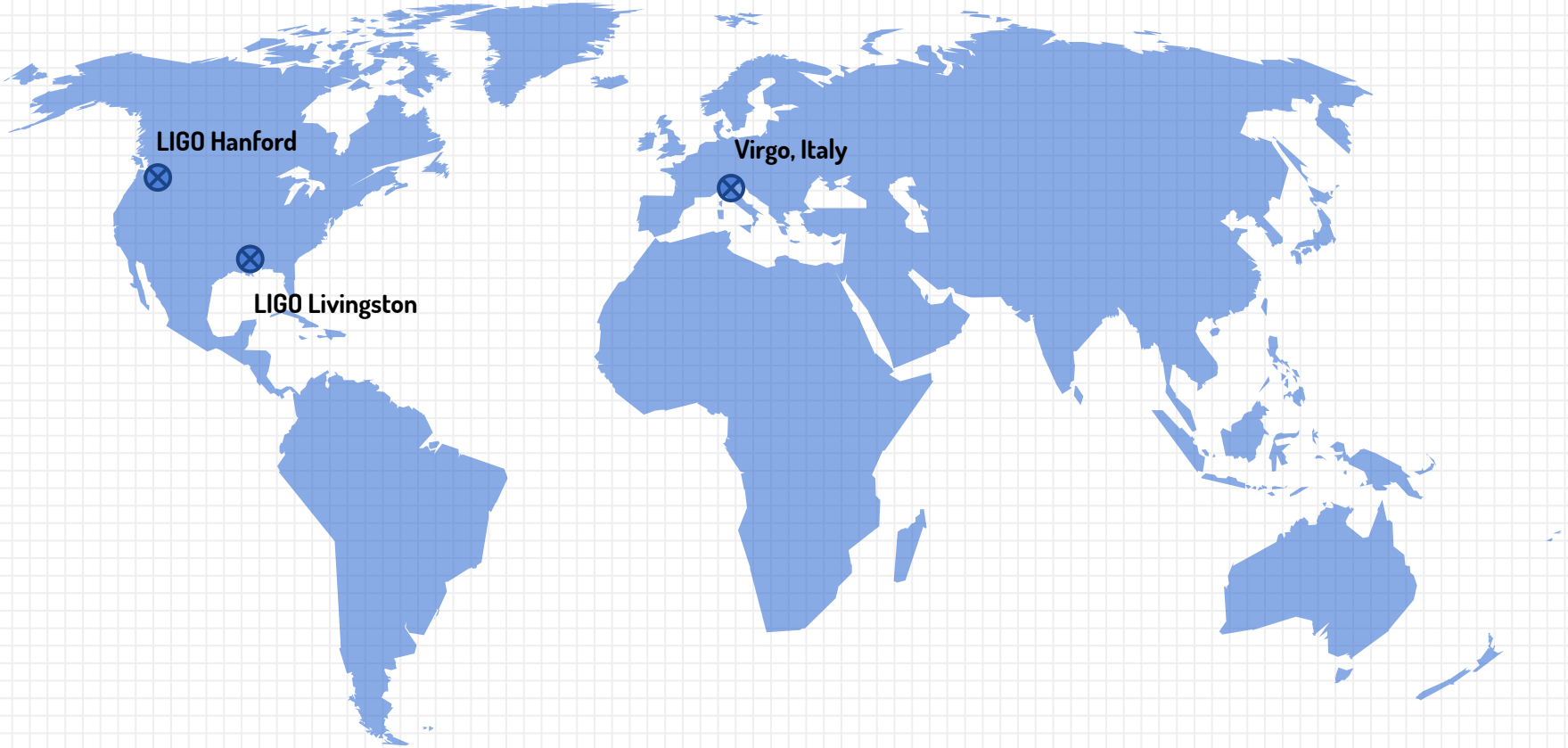
Build a Machine Learning pipeline to read, preprocess, train models and predict the gravitational wave signals & use ROC AUC metric to build the classifier.

How are GW Detected?

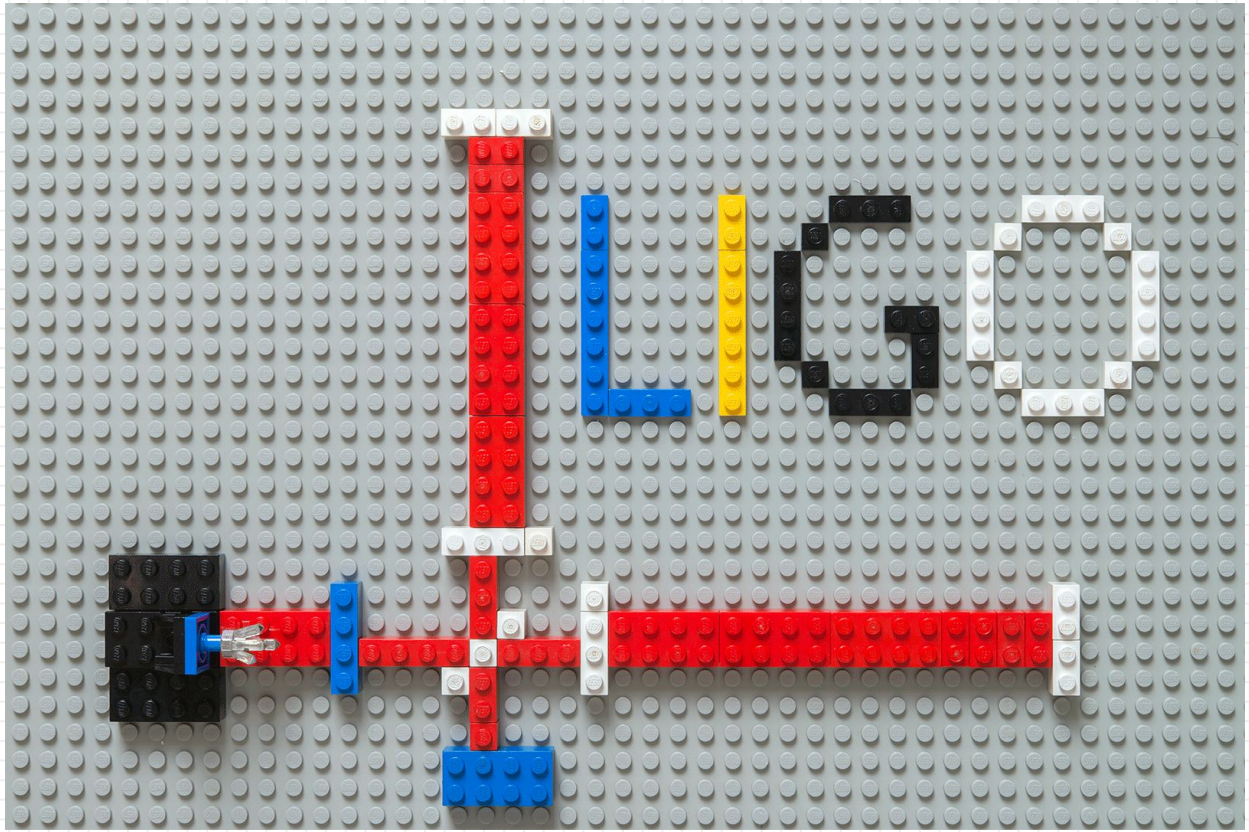
Anyone said LASER?



LIGO & Virgo Observatory locations



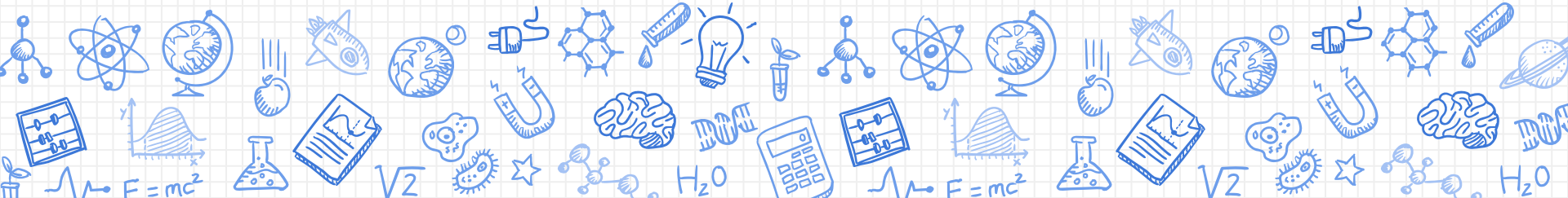
Laser Interferometers – strain



Source: https://futurumcareers.com/why-gravitational-waves-are-of-supermassive-importance

Look at Data

Challenges with Data Visualizations



72GB

That's a lot of data

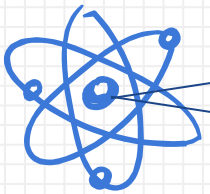
786,000 measurements

And a lot of observations

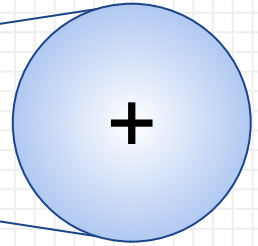
50~50

With balanced class distribution

How Small Did You Say it is?

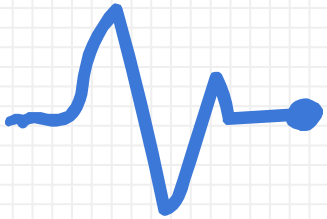


Atom



$\sim 10^{-15}$

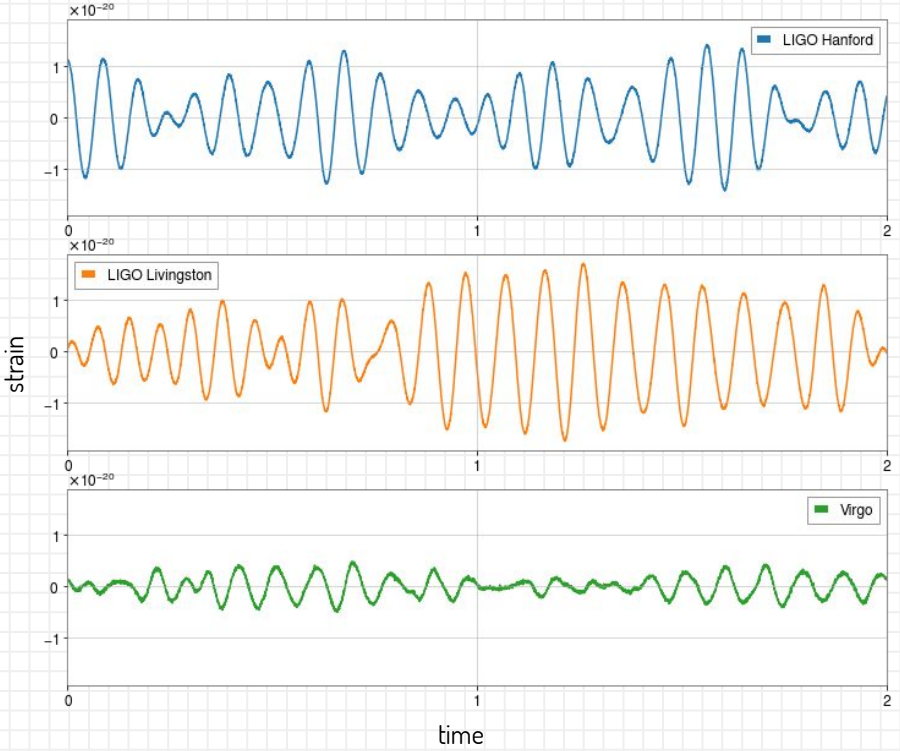
Black Hole



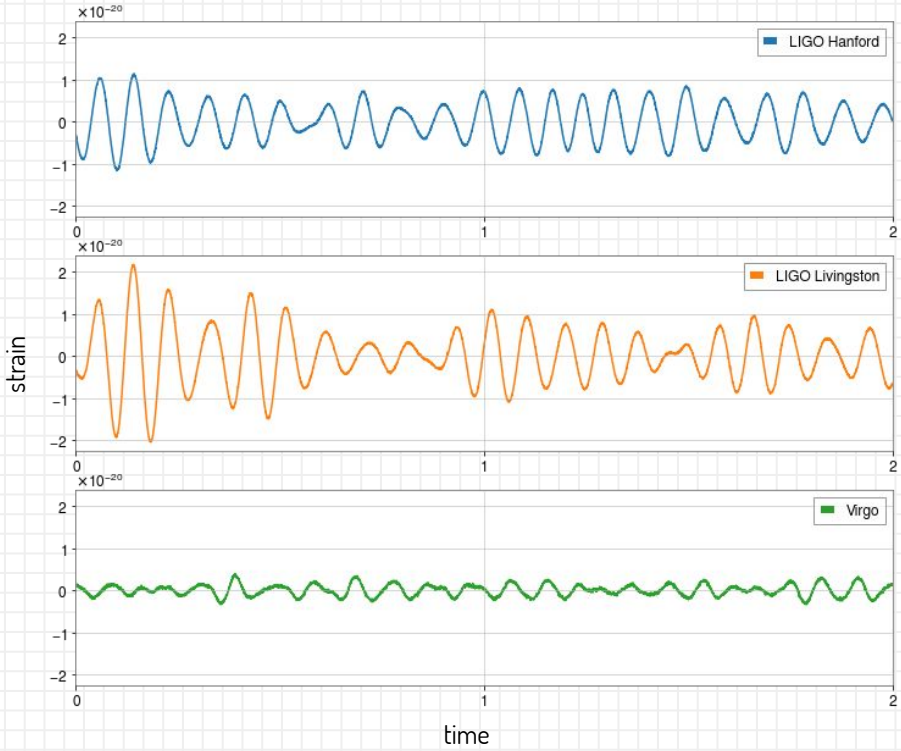
$\sim 10^{-20}$

Why do we need ML?

w/ GW



w/o GW



Spectrogram Transformation

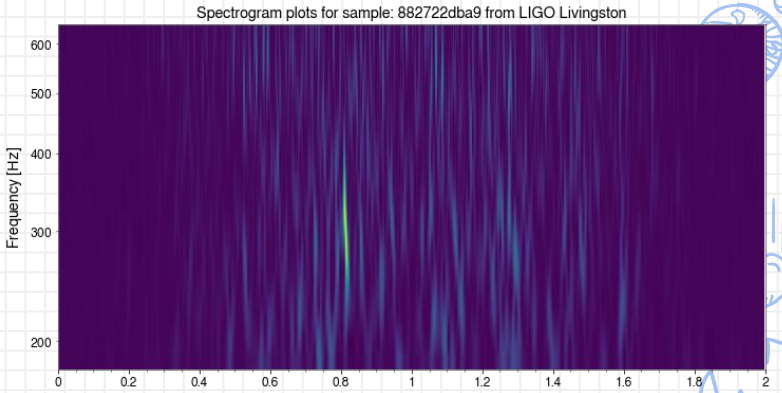
Time domain → Frequency domain

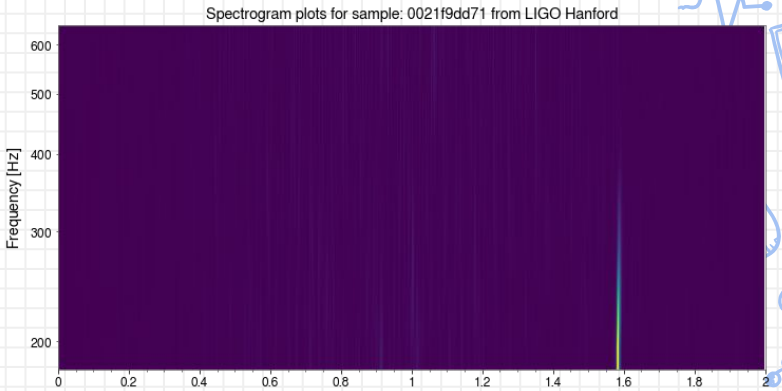
Signals = sine + cosine

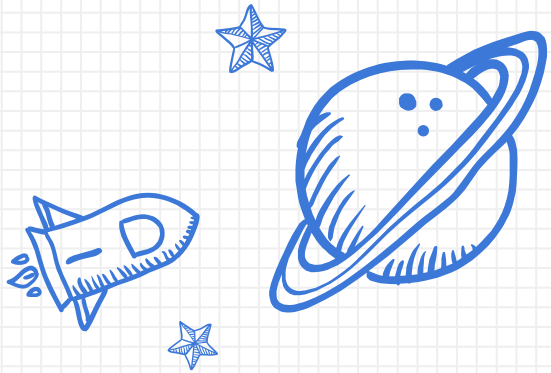
STFT: short term fourier transform

Changes in frequency over time removes unwanted white noise

Constant Q-Transform →







Modelling

Baseline CNN & SOTA Models

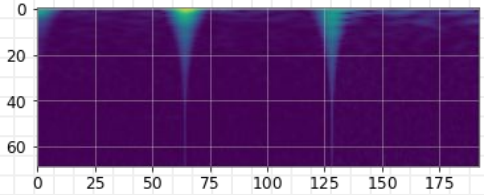
Building TensorFlow Input Data Pipeline

fetch

Read data
ID → path (.npy)

preprocess

np.load()
signal / np.max(signal)
np.hstack(signal)
Constant Q-Transform



batch

batch_size = 250
prefetch
shuffle train

Model Comparison

	CNN	EfficientNet
Architecture	3x Conv2D 3x MaxPool2D 1x Flatten 2x Dense	1x Conv2D 1x EfficientNet (imagenet) 1x GlobalAvgPool2D 1x Flatten 1x Dense
# Parameters	Total params: 4,382,081 Trainable params: 4,382,081	Total params: 4,090,619 Trainable params: 4,048,603 Non-trainable params: 42,016
Avg time/epoch	3300s / 55 min	6000s / 1 hr 40 min
Performance metrics	Train AUC: 0.83 I acc: 0.76 Valid AUC: 0.84 I acc: 0.77 kaggle: 0.843*	Train AUC: 0.82 I acc: 0.xx Valid AUC: 0.85 I acc: 0.xx kaggle: 0.857*

* Competition ongoing as of Aug 5, 2021

Conclusions

- **Big data** challenge addressed
- Achieved **0.86** AUC score on test with CNN
- Optimizing for GPU can run **5x** faster
- Train Once & predict forever (**50MB** model can predict **20GB** of data)
- Pre-trained weights of SOTA model converge faster

Future Work

Preprocess

> It makes a significant difference.

> See if using “raw” data to train the model is possible.

Compute Efficiency

> Take advantage of GPU.

> Make all functions & operations in TensorFlow pipeline GPU-compatible.

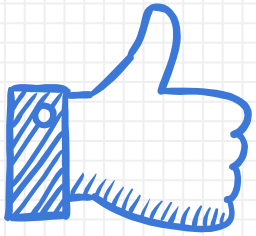
Regularization

> Train models with regularization to further improve performance.

Treat signals individually

> Three waves from each observation were stacked.

> Consider using three spectrograms to train model as multi-image input classifier.



THANKS!

Any questions?