ML and genetic algorithms for noise cancellation

NN subtraction with Neural Networks: an heuristic approach

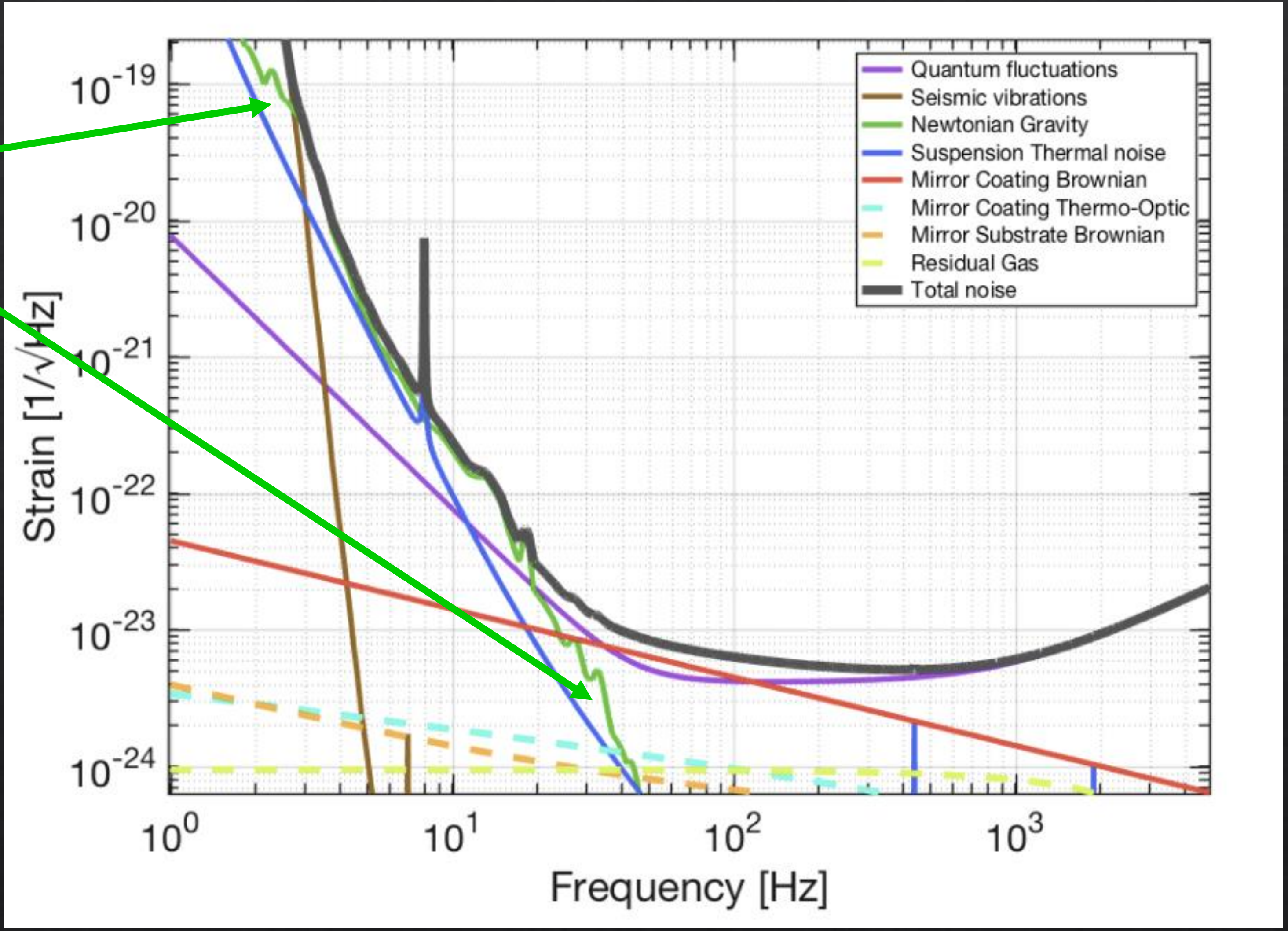
Alessio Cirone, Andrea Chincarini, Stefania Farinon

INFN Genova

1st Conference on Machine Learning for Gravitational Waves, Geophysics, Robotics, Control System and CA17137 MC2 meeting – WG3

Advanced Virgo noise budget

Newtonian Noise of seismic origin



Newtonian Noise

Seismic waves

Density fluctuations of the surrounding media (rocks, air, water…)

Gravity gradients

Newtonian Noise

Seismic waves

⇩

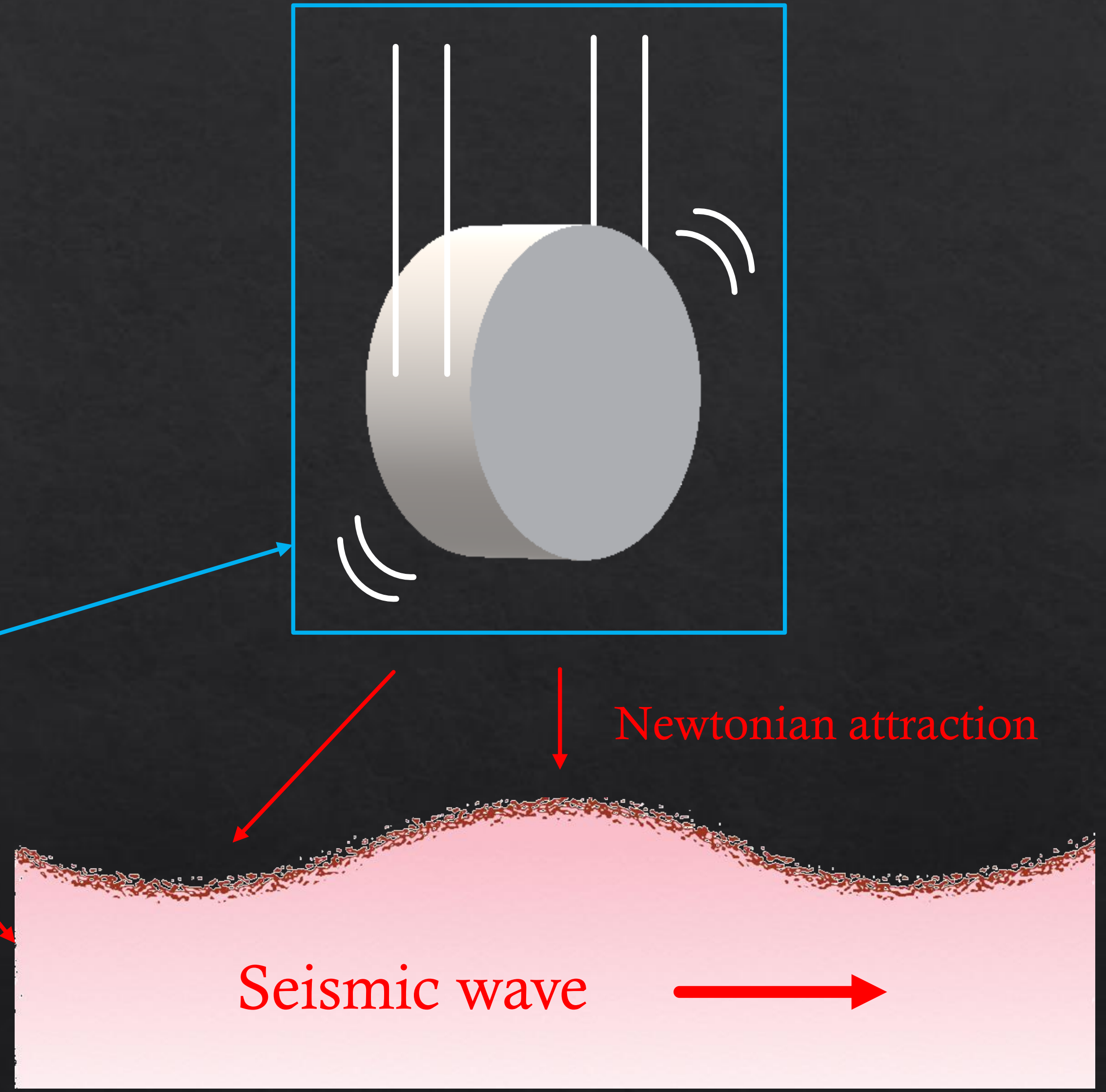
Density fluctuations of the surrounding media (rocks, air, water…)

⇩

Gravity gradients

⇩

Test Masses displacement



Newtonian Noise

Seismic waves

⇩

Density fluctuations of the surrounding media (rocks, air, water…)

⇩

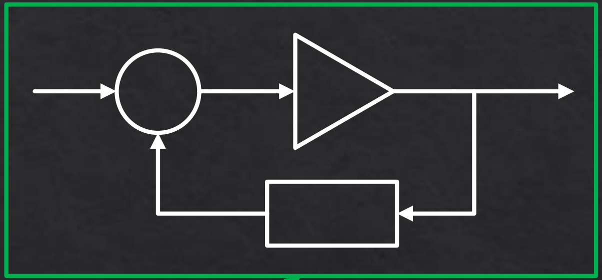
Gravity gradients

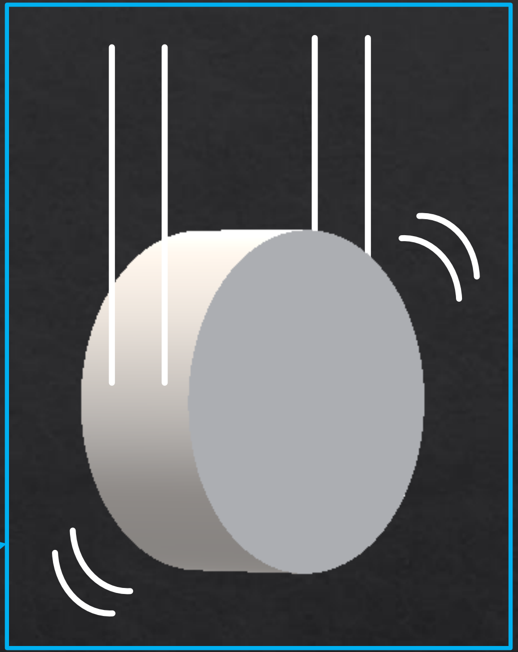
⇩

Test Masses displacement

⇩

Newtonian Noise (NN)

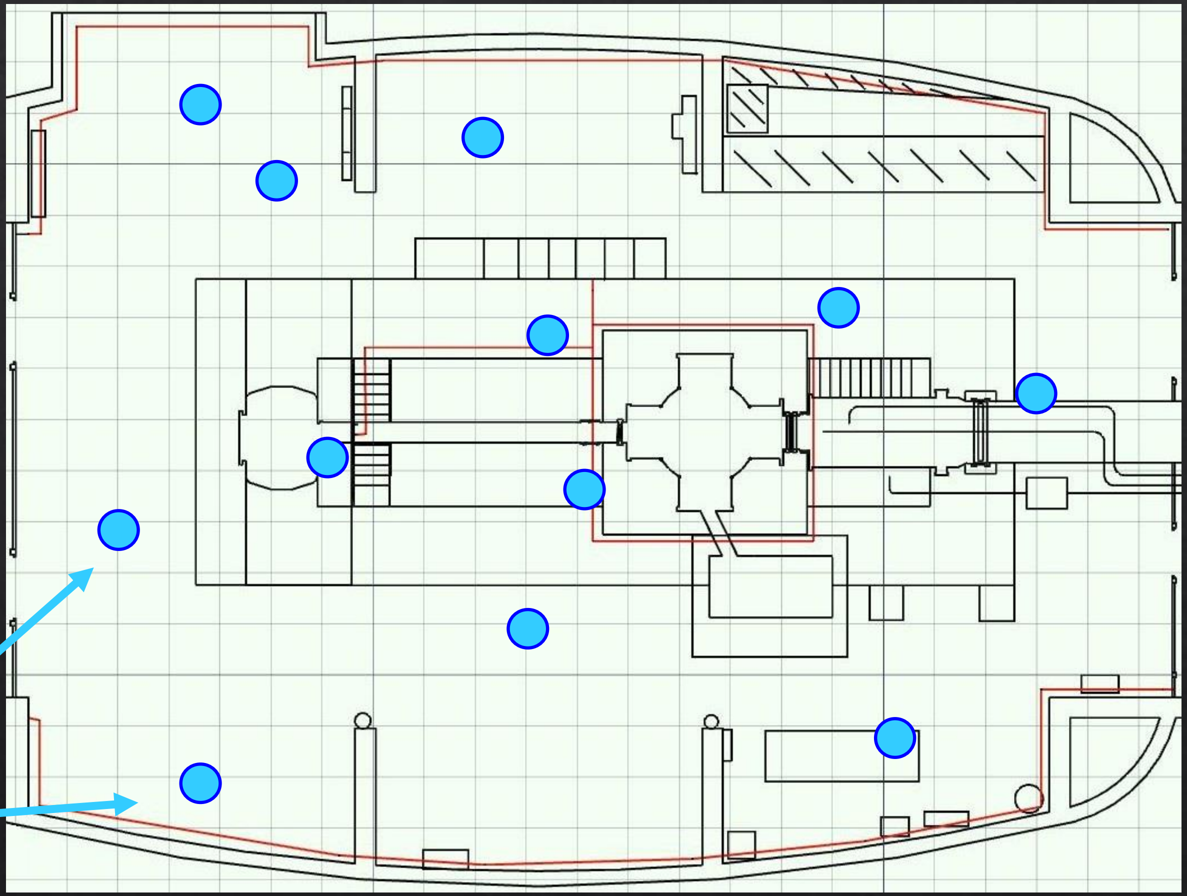




Problem definition

- Aim:
  - Predict NN-induced displacement on the test mass using an array of k heterogeneous seismic sensors
    - i.e. mix of accelerometers and tiltometers
    - With sub-optimal positioning
    - Open to future sensors extension

North End Building

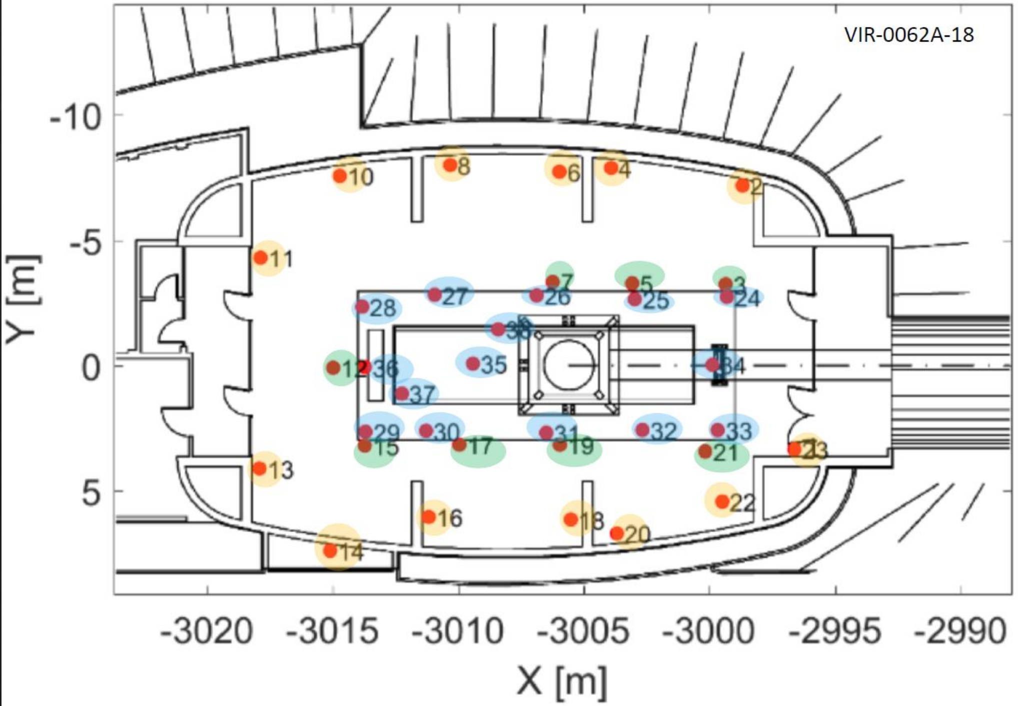


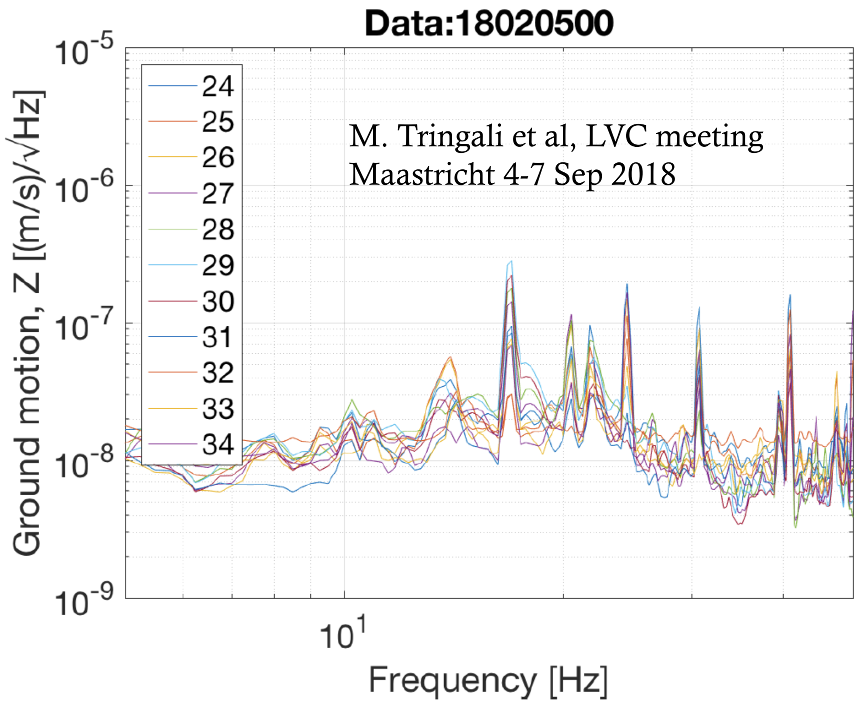
Redundant sub-optimal array

State of the art

- 38 seismic sensors, deployed on January 2018 inside the West End Building

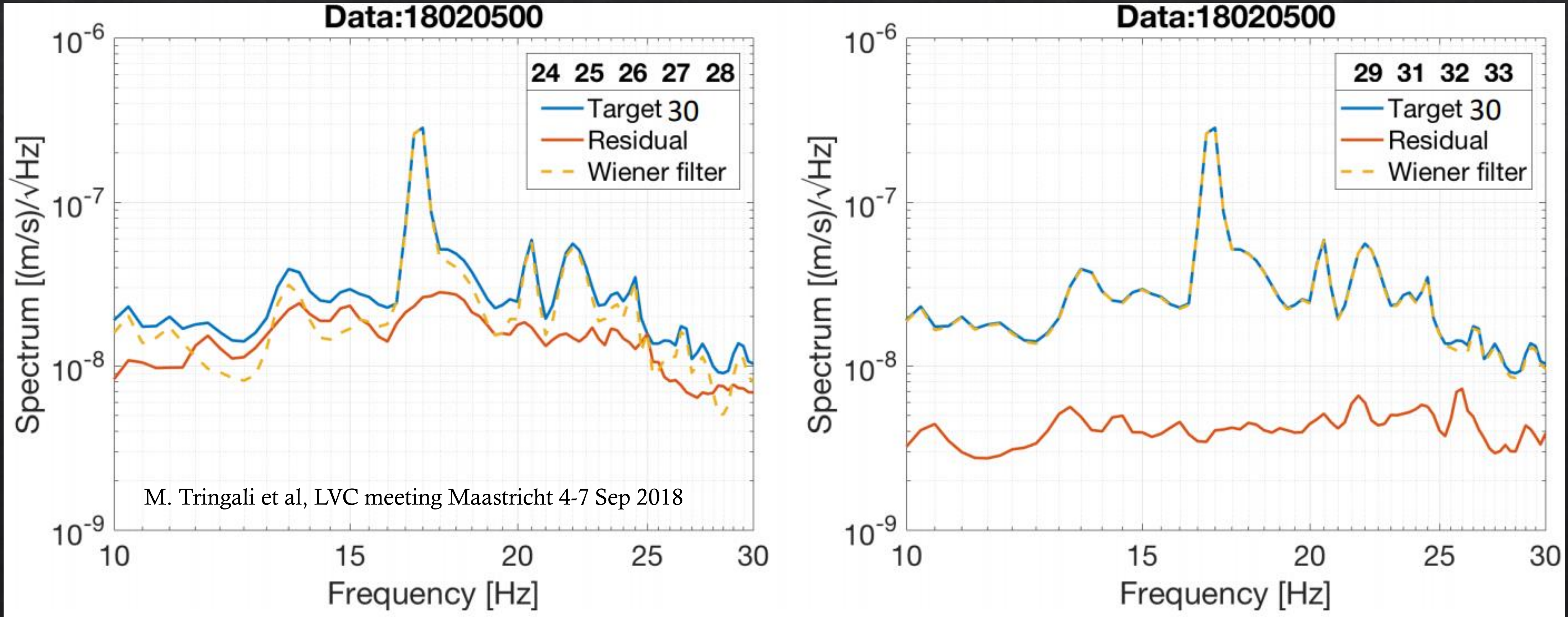
- Near walls
- Near tower platform
- On the tower platform





State of the art

Wiener filtering: sometimes high and other times poor coherence between filtered and target channels
Strong dependency on sensor selection!



Problem definition

NOW

- No real noise substraction yet (the target channel is still another seismic sensor instead of the test mass)
- Dependency on infrastructure and soil properties
- Focus on sensor array disposition

- Proposed solution:
  - Use a neural network that is:
    - Robust with respect to sensors placement, number and type
    - Robust with respect to terrain parameters
    - Able to predict Newtonian forces on the test mass better than Wiener filtering (gold standard)
  - Yes but … which architecture?
  - How to train the network?
  - How to evaluate the robustness and the extensibility?

Activity plan

- How do we choose the correct network architecture?

- Proposed method:
    - Evolve possible architectures on simulated data (Finite Element Analysis)
    - Use Genetic Algorithms (GA)
    - Exploit the parallel nature of GA on our dedicated computing farm
    - Try ML on real displacement data

Which data?

- Experimental seismic evaluation, done last year by the Polish team, at the West End Building (25 Jan – 6 Feb):
  - Array of about 40 seismometers
  - Sampling frequency 1 kHz
  - Almost 50 GB of data

- Our data processing:
  - Sub-sample of a few hours of good data
  - Resampling at 200 Hz
  - Band-pass filtering at [5-60] Hz

- We have enough data to train a Deep Neural Network

- What we miss (due to current Virgo sensitivity) is the «real» Newtonian Noise (i.e. DARM correction to the mirror due to NN)

The big question

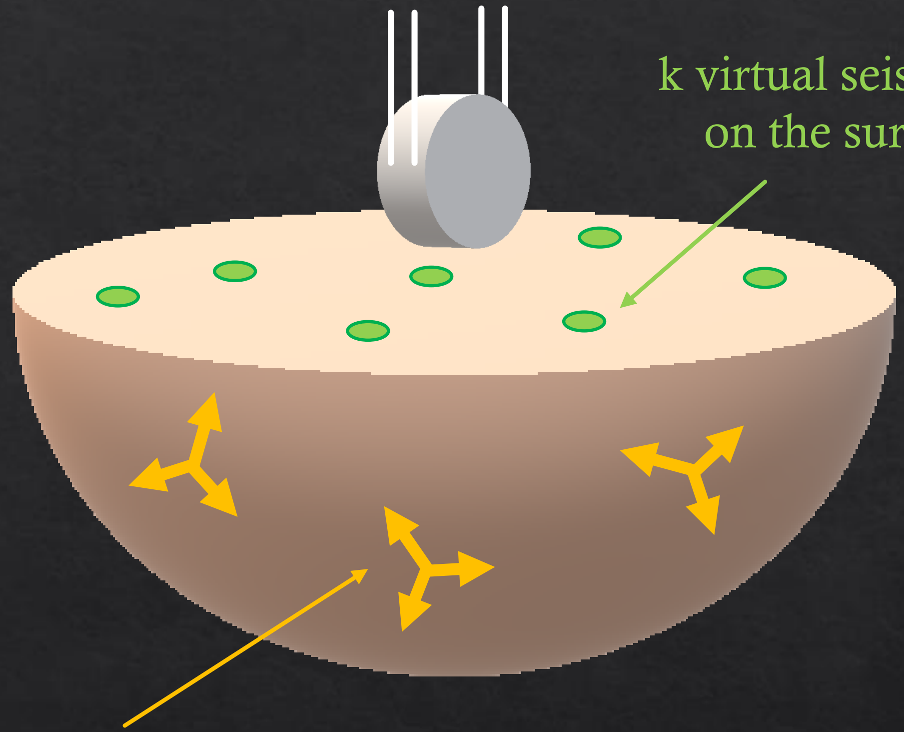
Up to what point can we find equivalence between predicting one sensor output (by knowing all the others) and directly predicting the Newtonian noise on the test mass?

We can do both cases with simulated data!

If we achieve good prediction on simulated data, then we will be able to calculate a good estimate of the error on real data

Basic simulation structure

- Simulate seismic waves in a multi-layered terrain
  - Build a simple case study
    - Homogeneous half-sphere
    - N point sources
    - Compute the resulting force on the test mass (1 m high), from the displacements calculated by k virtual seismic sensors on the surface

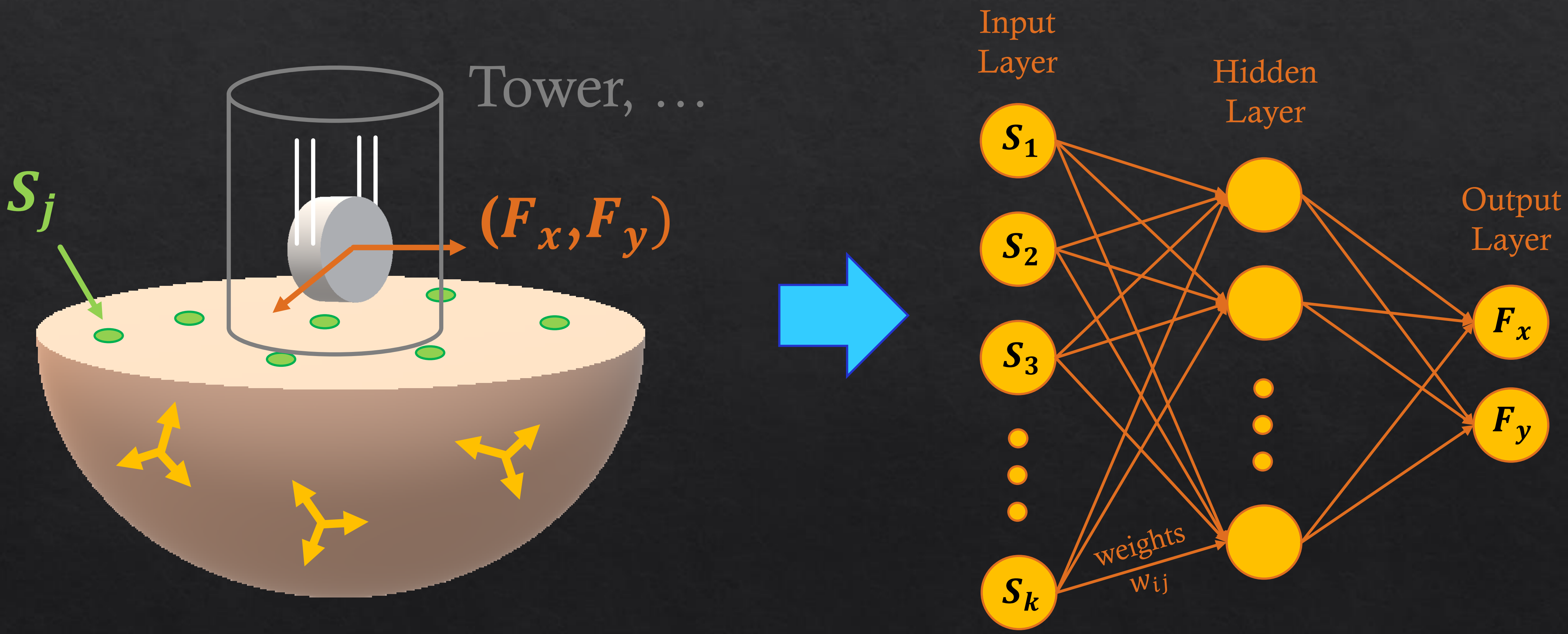


k virtual seismometers on the surface (S_j)

N point sources randomly distributed: $P_i = f_0 \sin(\omega_i t + \varphi_i)$
With $i = [1, N]$,
φ random,
$\omega = 2\pi f$ with $f = [5{,}40]$ Hz,
$f_0 = 1kN$

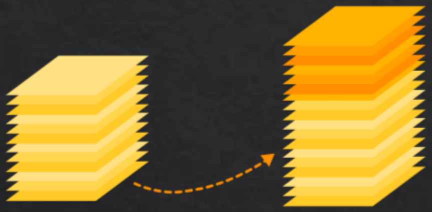
Basic simulation structure

- Compare various models: from simple to complex ones
  - Semi-infinite solid, multi-layered terrain, effect of buildings and infrastructures
- Test various configurations for displacement measurements (number of sensors, positioning …)

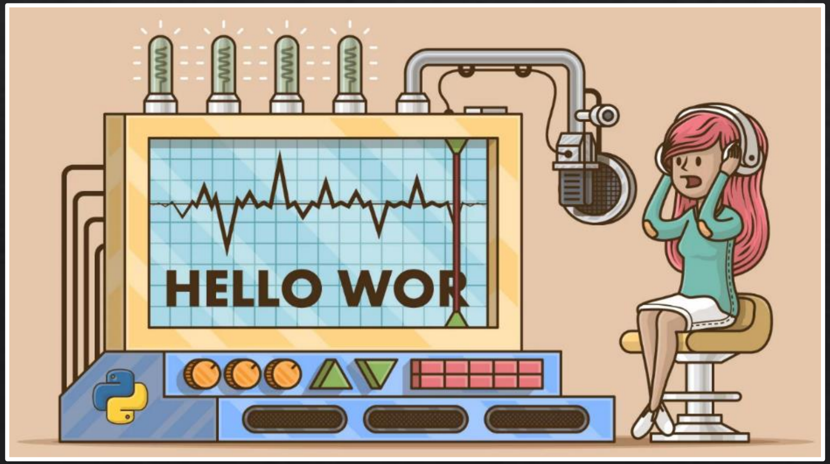


Neural Network

- The Network takes the surface displacements as input and the forces on the test mass as output



time	Sensor 1	Sensor 2	Sensor 3	…	Sensor k	Force x	Force y
0.01	1,24E-11	1,26E-13	3,32E-11	…	2,72E-14	-2,30E-07	3,03E-08
0.02	3,16E-10	5,31E-13	6,95E-10	…	9,48E-13	5,11E-08	1,71E-07
0.03	3,66E-09	4,03E-13	8,48E-09	…	1,57E-11	-4,68E-07	6,17E-07

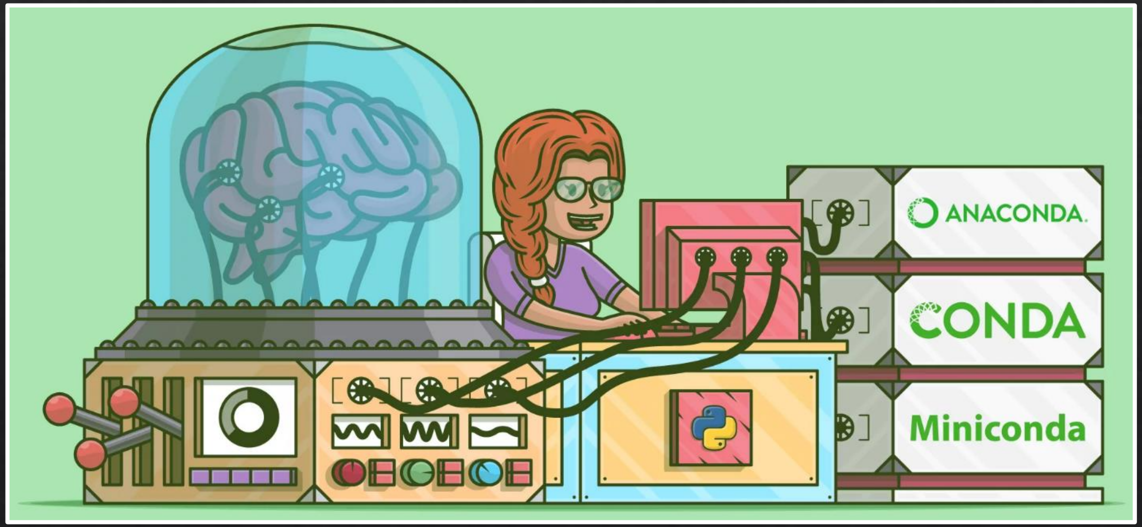


- The Network is trained and optimized on the simulated data (process done on all the CPUs in parallel)

Computing infrastructure

- We have at our disposal a small test cluster of about 140 cores for distributed processing

- We have set a virtual environment with Anaconda python distribution



Computing infrastructure

We use the Python based Keras library, the TensorFlow backend and some extra packages like SCOOP (Scalable COncurrent Operations in Python) and DEAP (Distributed Evolutionary Algorithms in Python).

Neural Network architecture

We build a Sequential model, which is just a linear sequence of Fully Connected, Dropout, Convolutional … layers.

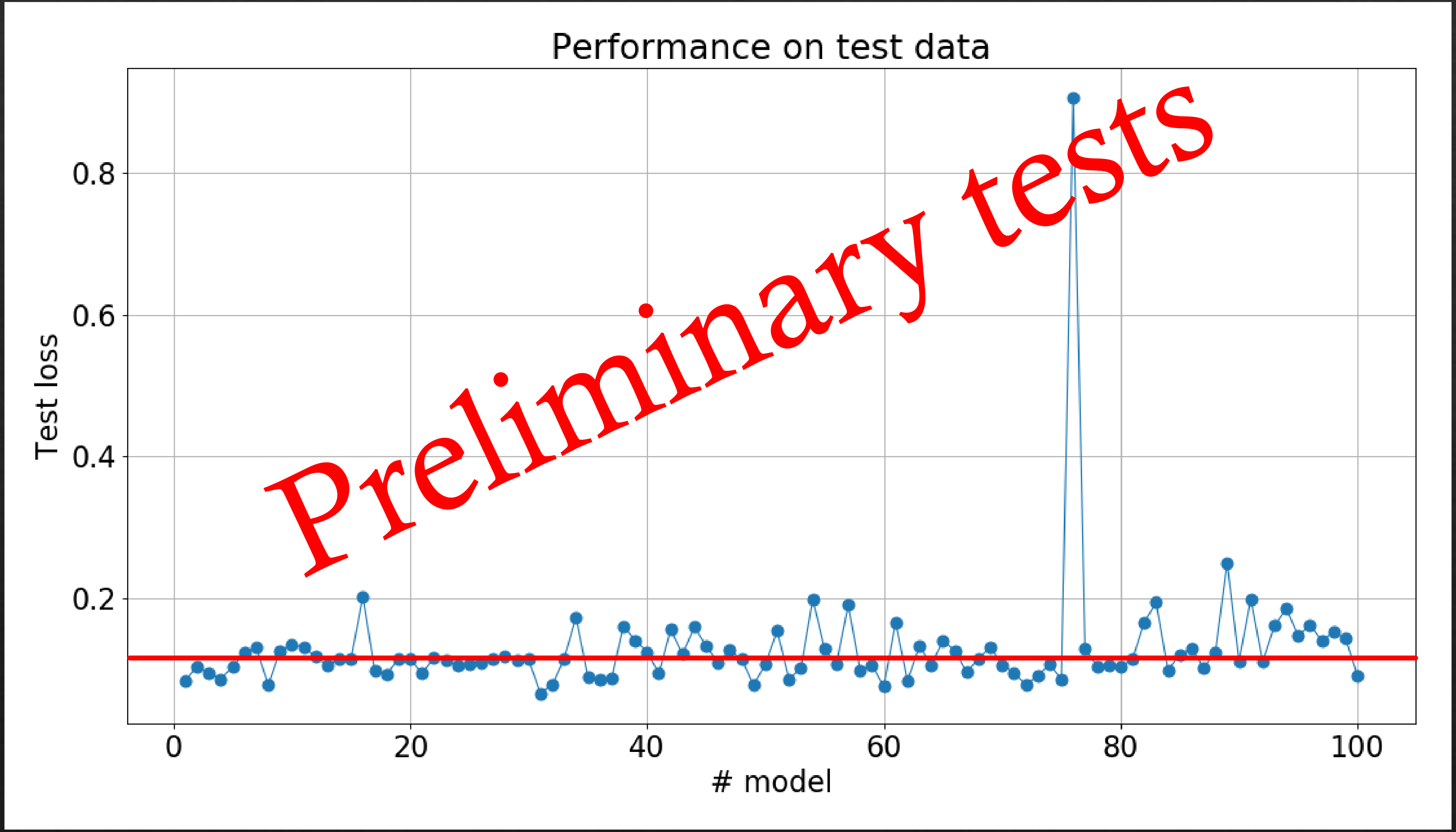
Then the model is compiled with some particular instructions:

- Error function: it tells how to evaluate the performance of the network. In our case it is the Mean Squared Error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i^{pred} - y_i^{true}\right)^2$$

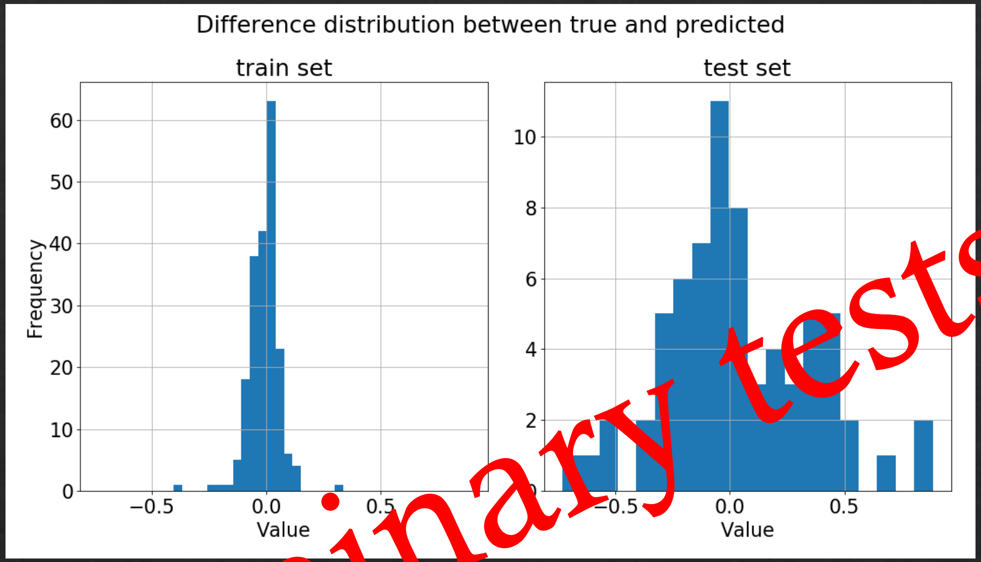
- Optimization algorithm: it helps to minimize the error function in order to produce slightly better and faster results by updating the model parameters such as weights and biases

Evaluate the performance: only one generation

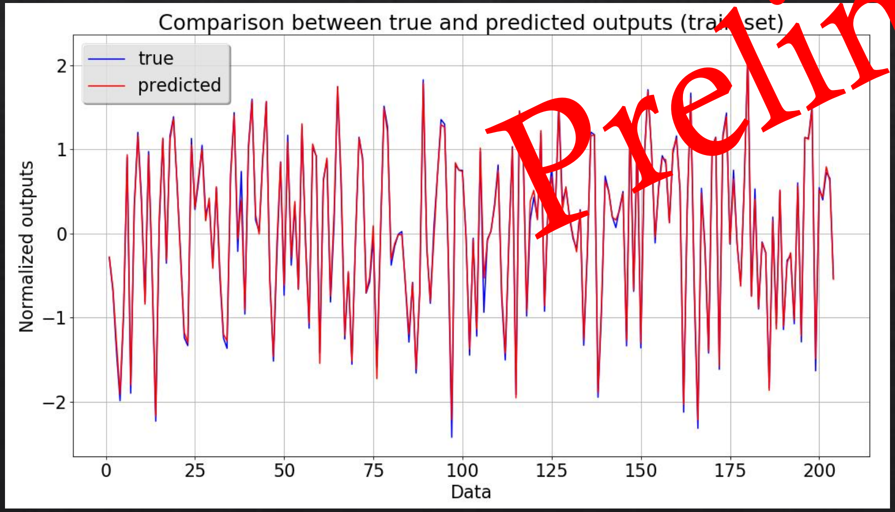


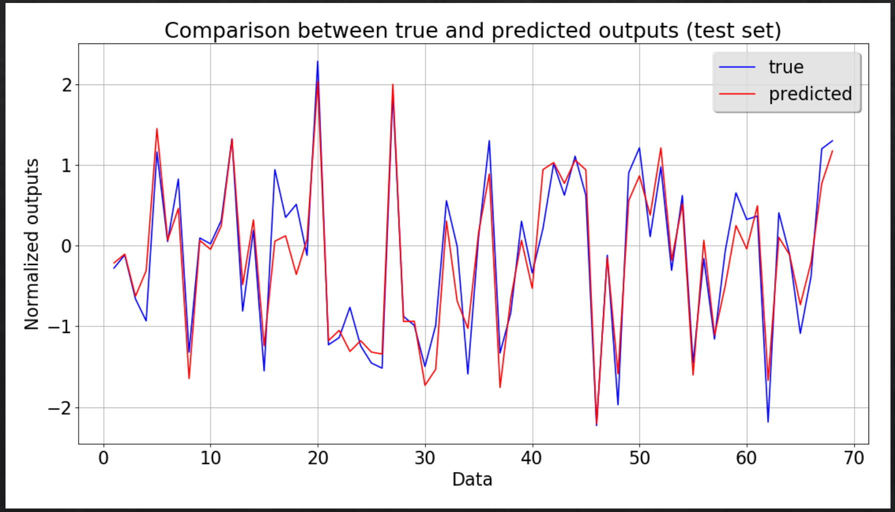
- Supervised learning on 100 networks with different architectures (number of neurons per layer and number of layers), with an average performance of 10%
- Not enough to find the best network structure

Evaluate the performance: only one generation



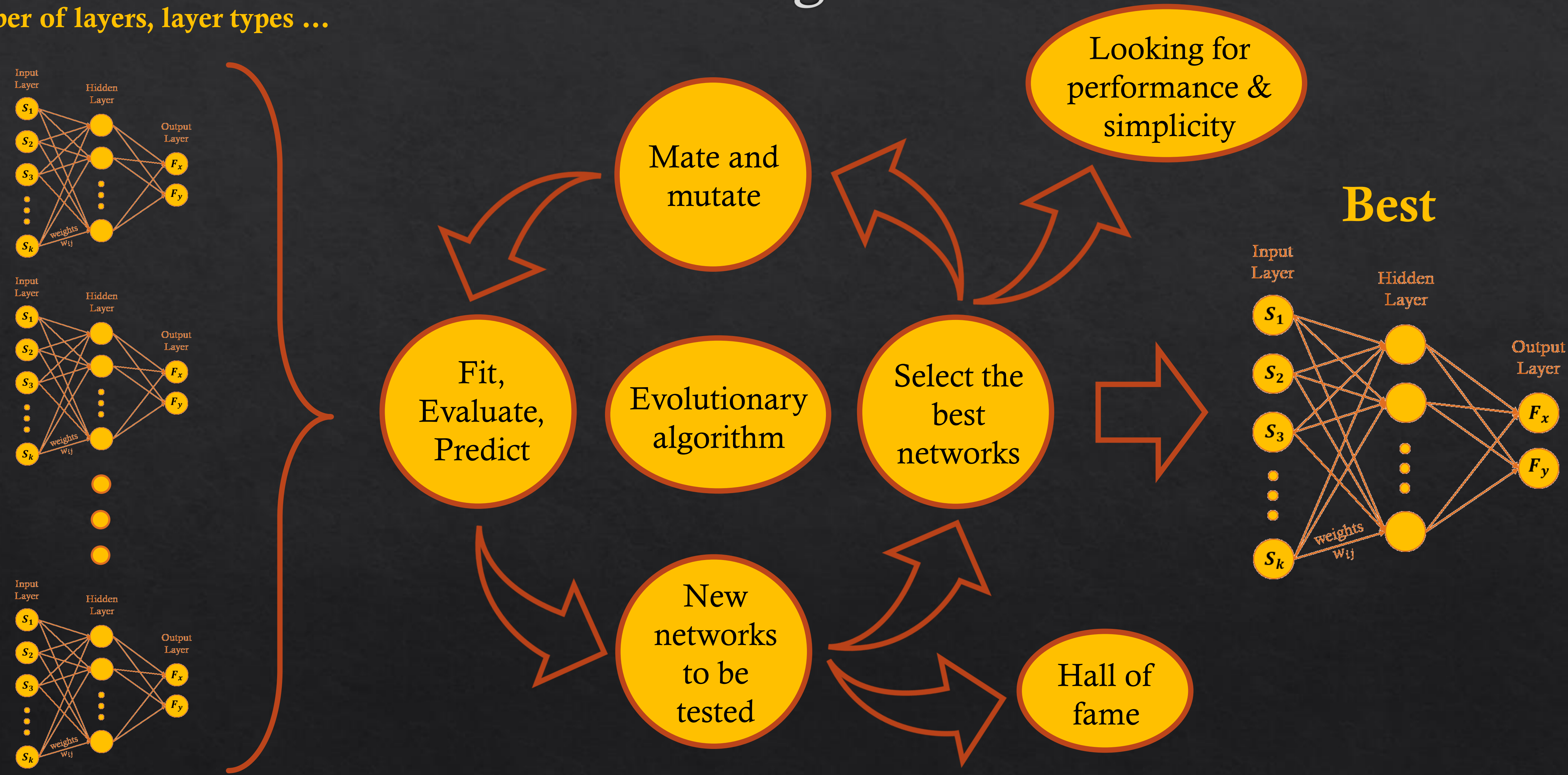
Preliminary tests





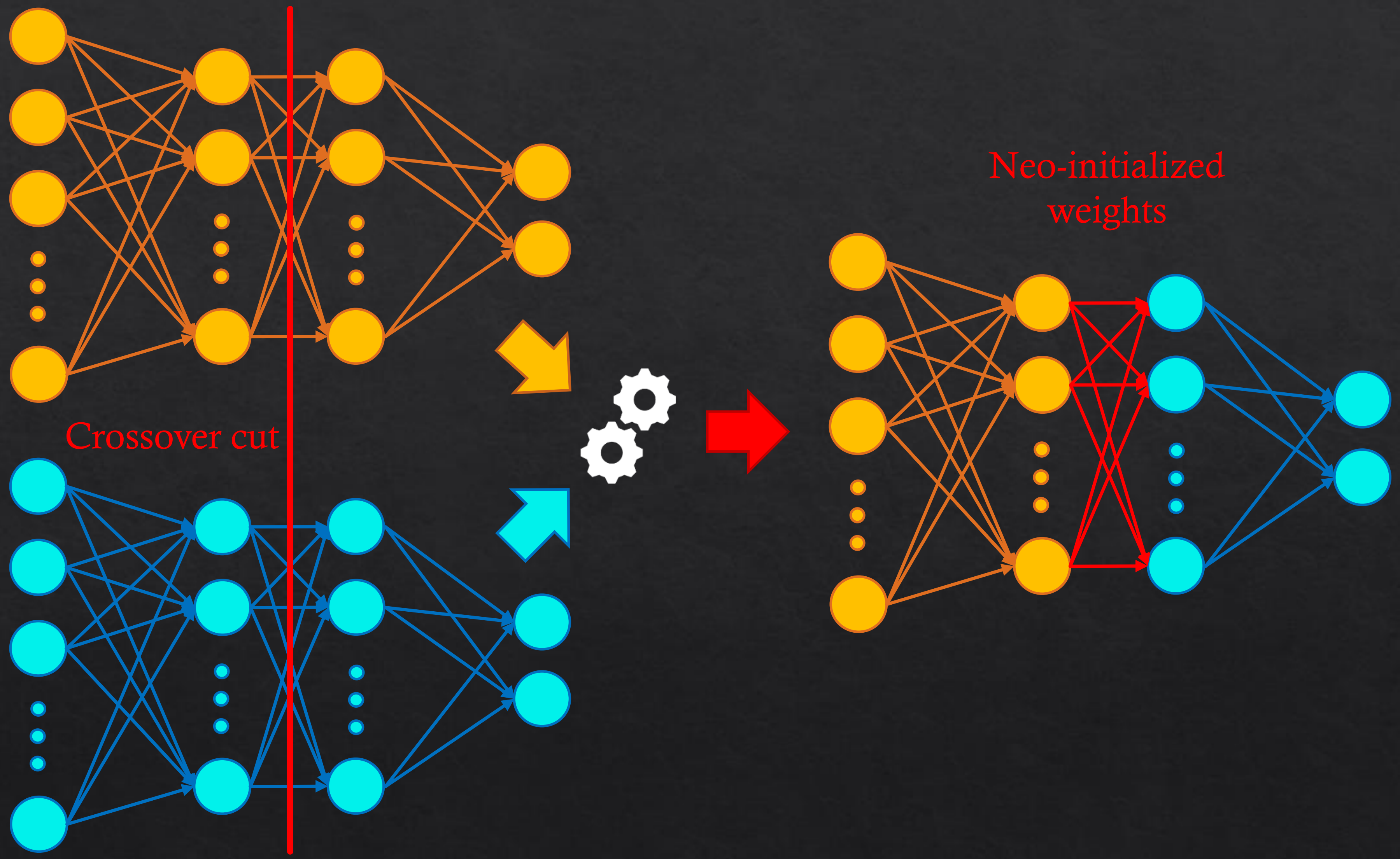
Genetic algorithm

Variable number of neurons, number of layers, layer types …

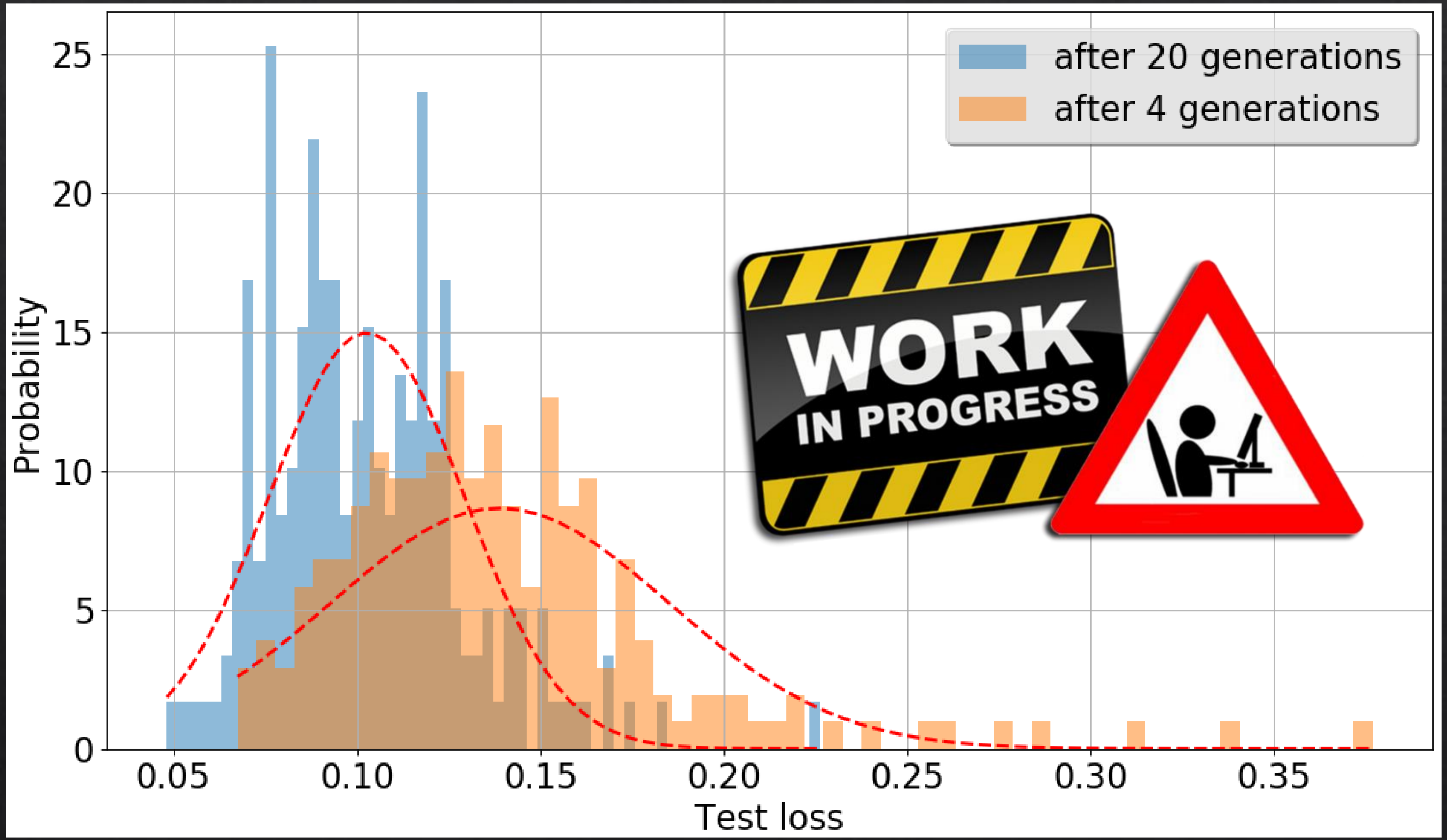


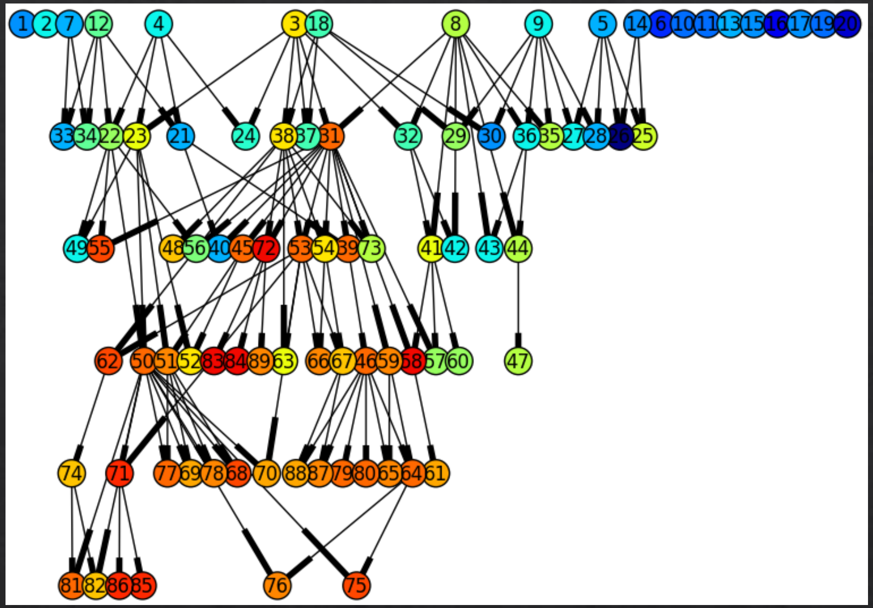
Transfer learning to lower the computation time

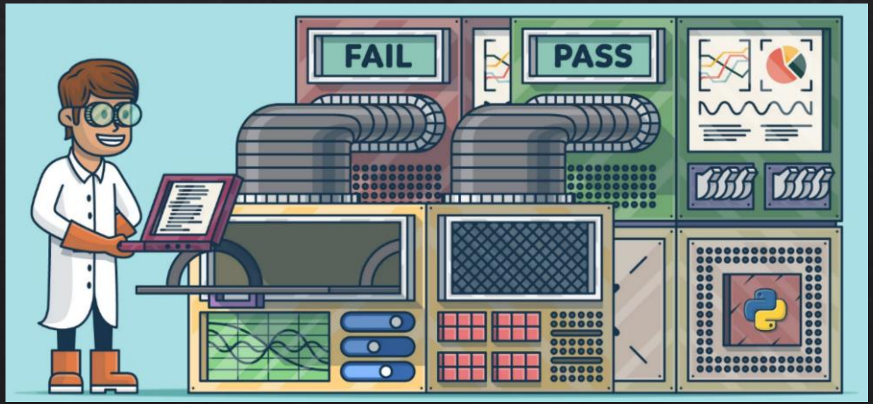
- In GA new networks are created by "mating" two high-performance networks
- The new networks inherit some properties from both parents, in particular their weights, therefore applying an heuristic transfer learning



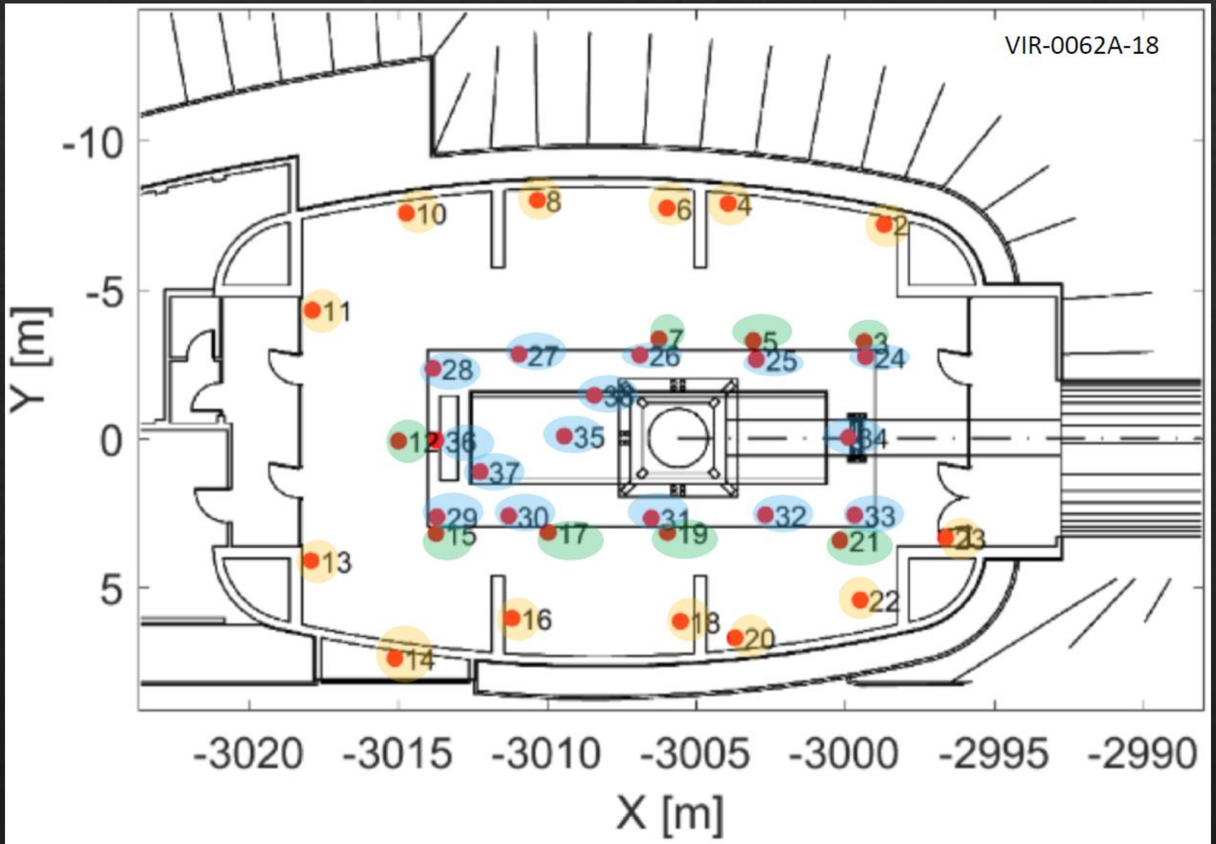
Evaluate the performance







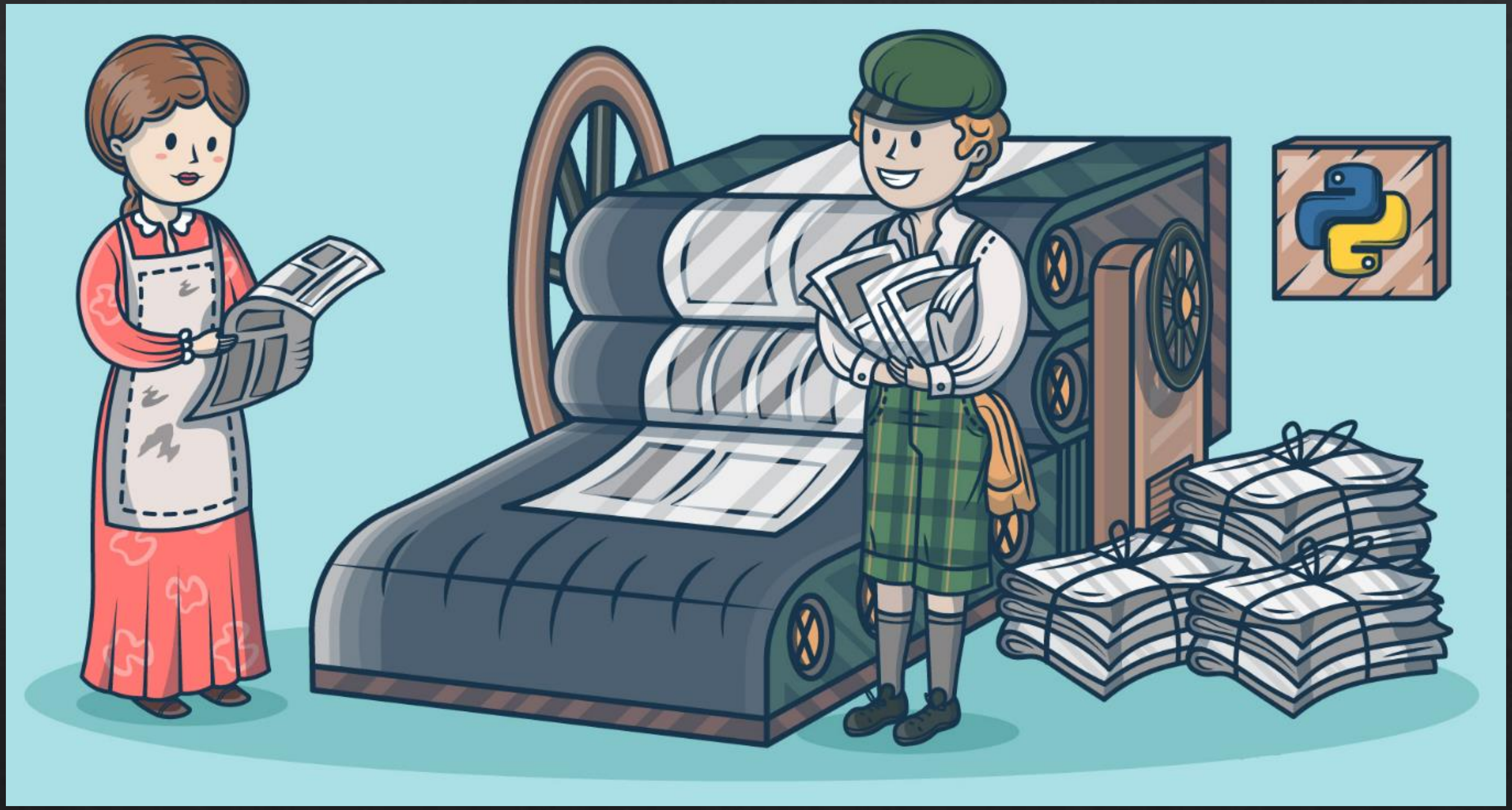
Outlook – real seismic data



- Displacement data were gently provided by Jan Harms
- In this case we will use a target sensor as output, instead of the forces on the test mass, which is done also with Wiener filtering
- The goal is to reproduce the Wiener filter substraction performances and try to improve them

Portability

In the future we would like to build a robust portable environment for Virgo to run in loco and act offline for Newtonian noise substraction

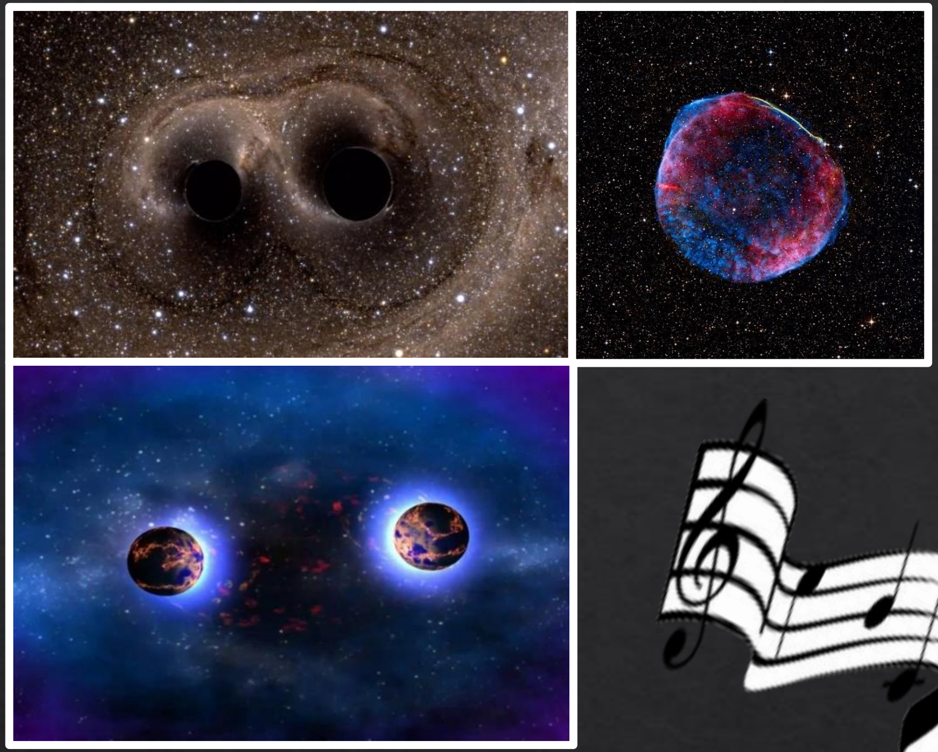


Good foundations:

- python based
- open access
- CPU & GPU friendly

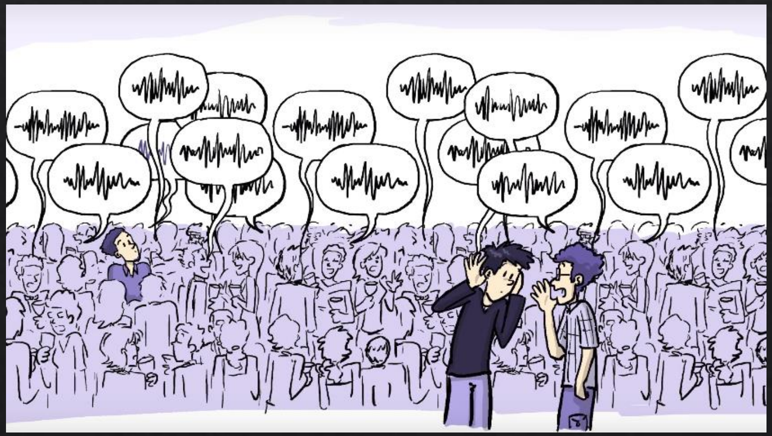
The end

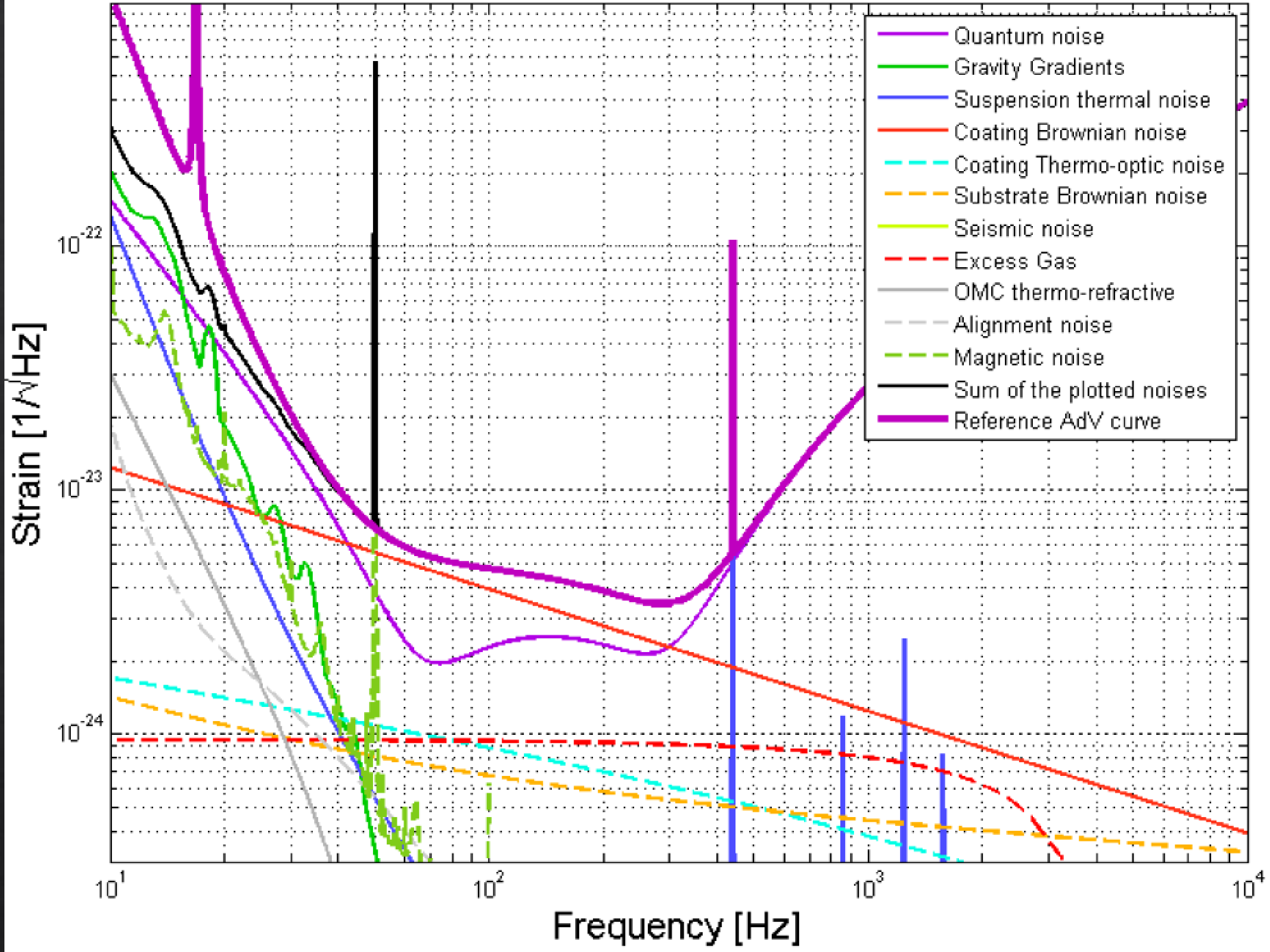
Back up slides



Recap: Gravitational Waves

Advanced Virgo is a laser interferometer devoted to the detection of Gravitational Waves of astrophysical and cosmological origin. The detection power lies entirely on the instrument spectral sensitivity.




Quantum noise
Gravity Gradients
Suspension thermal noise
Coating Brownian noise
Coating Thermo-optic noise
Substrate Brownian noise
Seismic noise
Excess Gas
OMC thermo-refractive
Alignment noise
Magnetic noise
Sum of the plotted noises
Reference AdV curve
Strain [1/√Hz]
10^{-22}
10^{-23}
10^{-24}
10^{1}
10^{2}
10^{3}
10^{4}
Frequency [Hz]