Finanziato dall'Unione europea NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Neural network time-series classifiers for GW searches in single-detector periods

A. Trovato* , E. Chassande-Mottin, M. Bejger, R. Flamary, N. Courty,

*Università di Trieste, INFN-Sezione Trieste

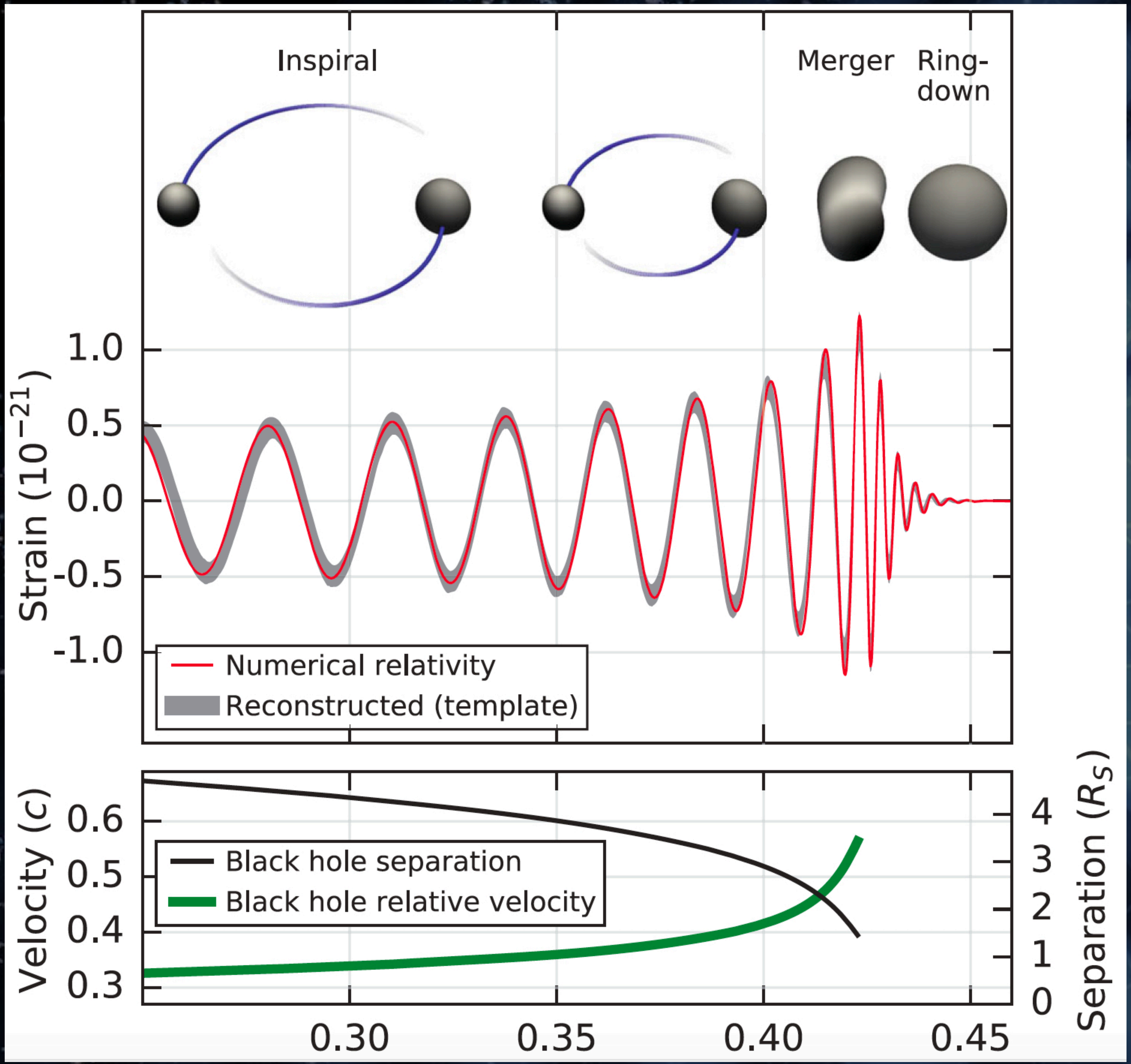
UNIVERSITÀ DEGLI STUDI DI TRIESTE

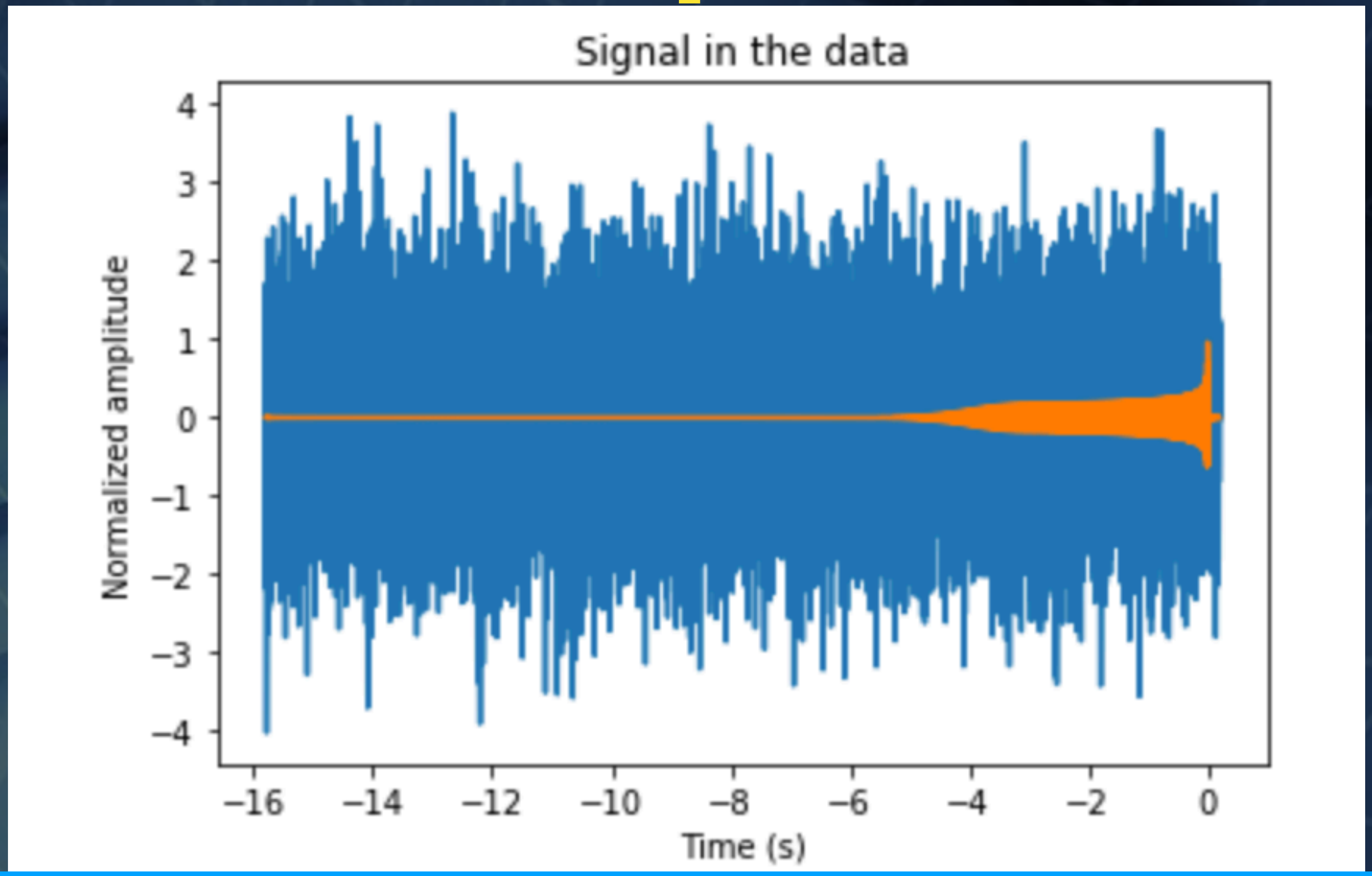
Dipartimento di Fisica

Dipartimento d'Eccellenza 2023-2027

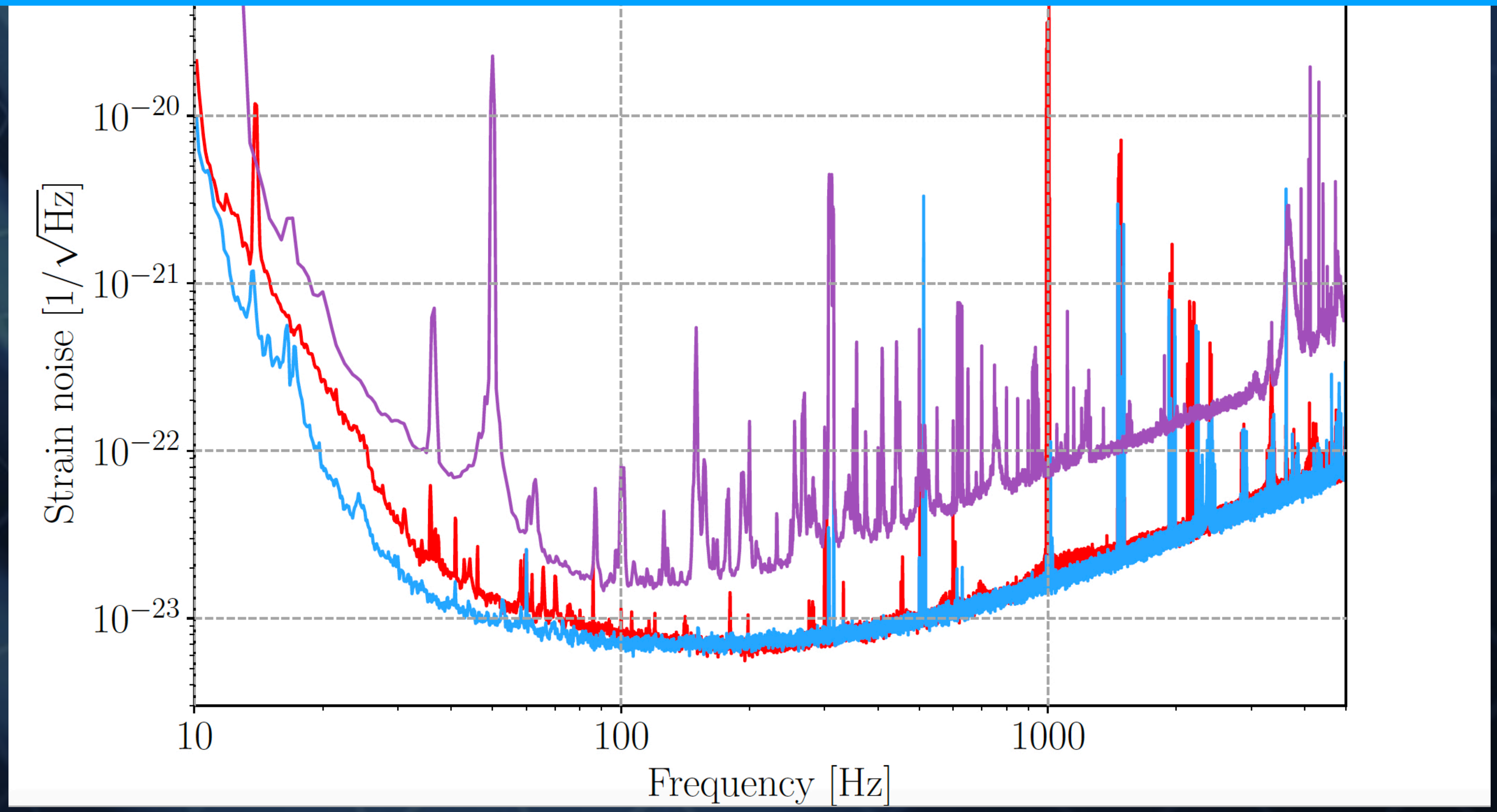
INFN Istituto Nazionale di Fisica Nucleare

Gravitational waves detection problem

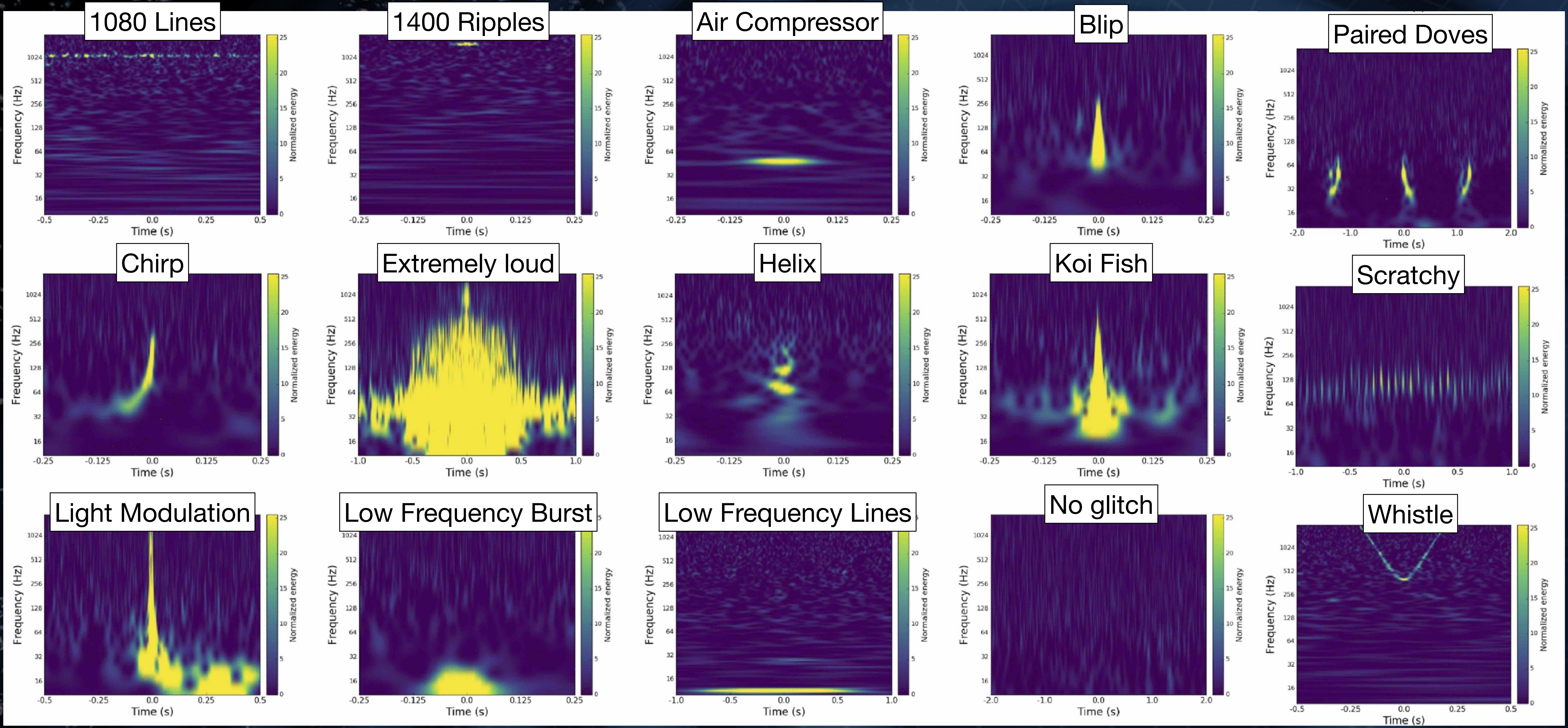




Rare and weak signals in complex background: non-Gaussian non-stationary

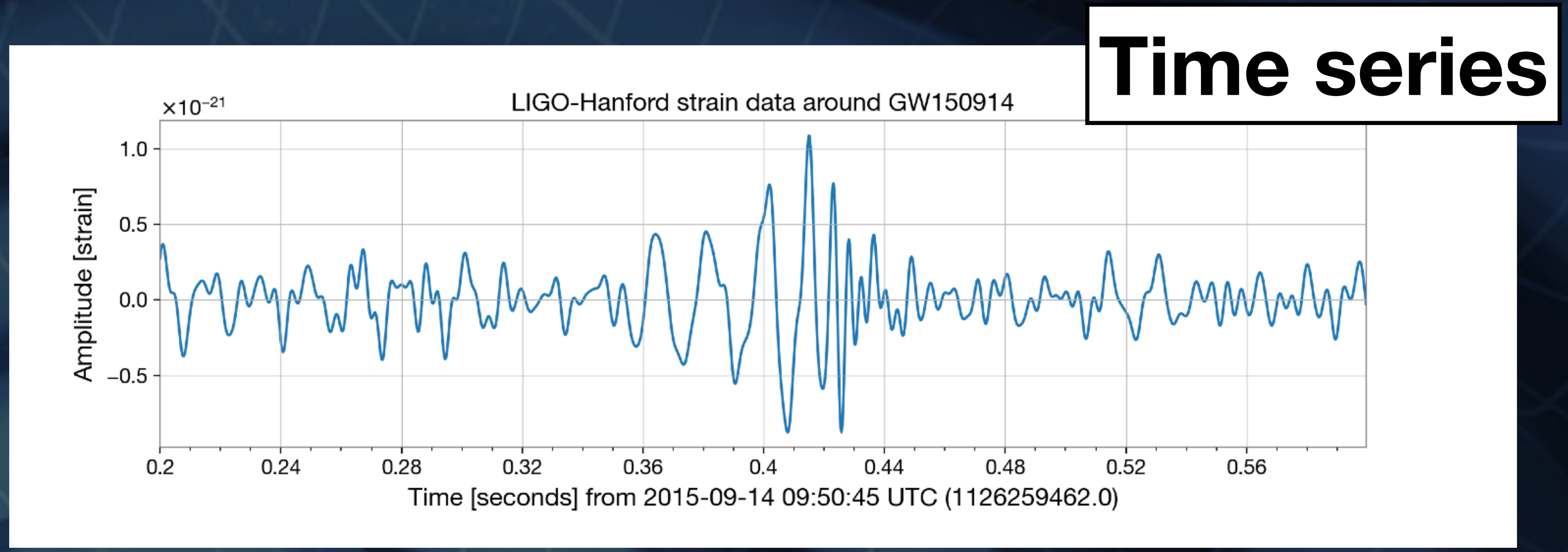


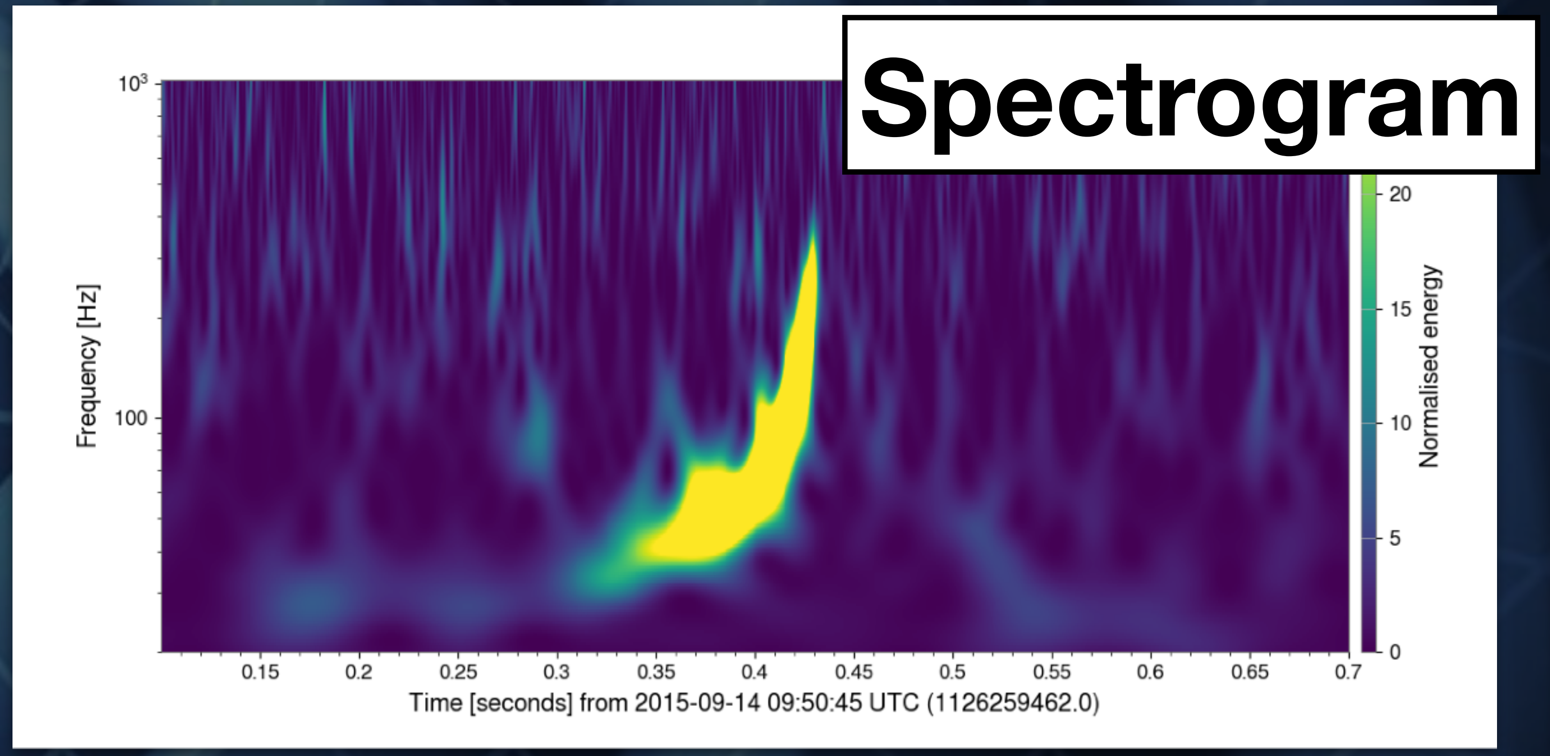
Glitches zoo



⋆**Credits: Gravity Spy dataset**

Work presented here





- ✓ Classification of segments of data
- ✓ Time-series representation
- ✓ Training on real data
- ✓ Focus on single detector periods
- ✓ Analysis of L1 single detector periods in O1
  - Previous works aim mainly at multi-detector analysis
- ✓ Paper available at: A Trovato et al 2024 Class. Quantum Grav. 41 125003

Single-detector time

- Glitch impact on sensitivity is larger during single-detector periods as coincidence with additional detector is impossible. Can machine learning help?
- Single-detector time:
  - ✓ ~2.7 months in O1+O2; ~1.6 months in O3: ~ 2.4 months in O4a

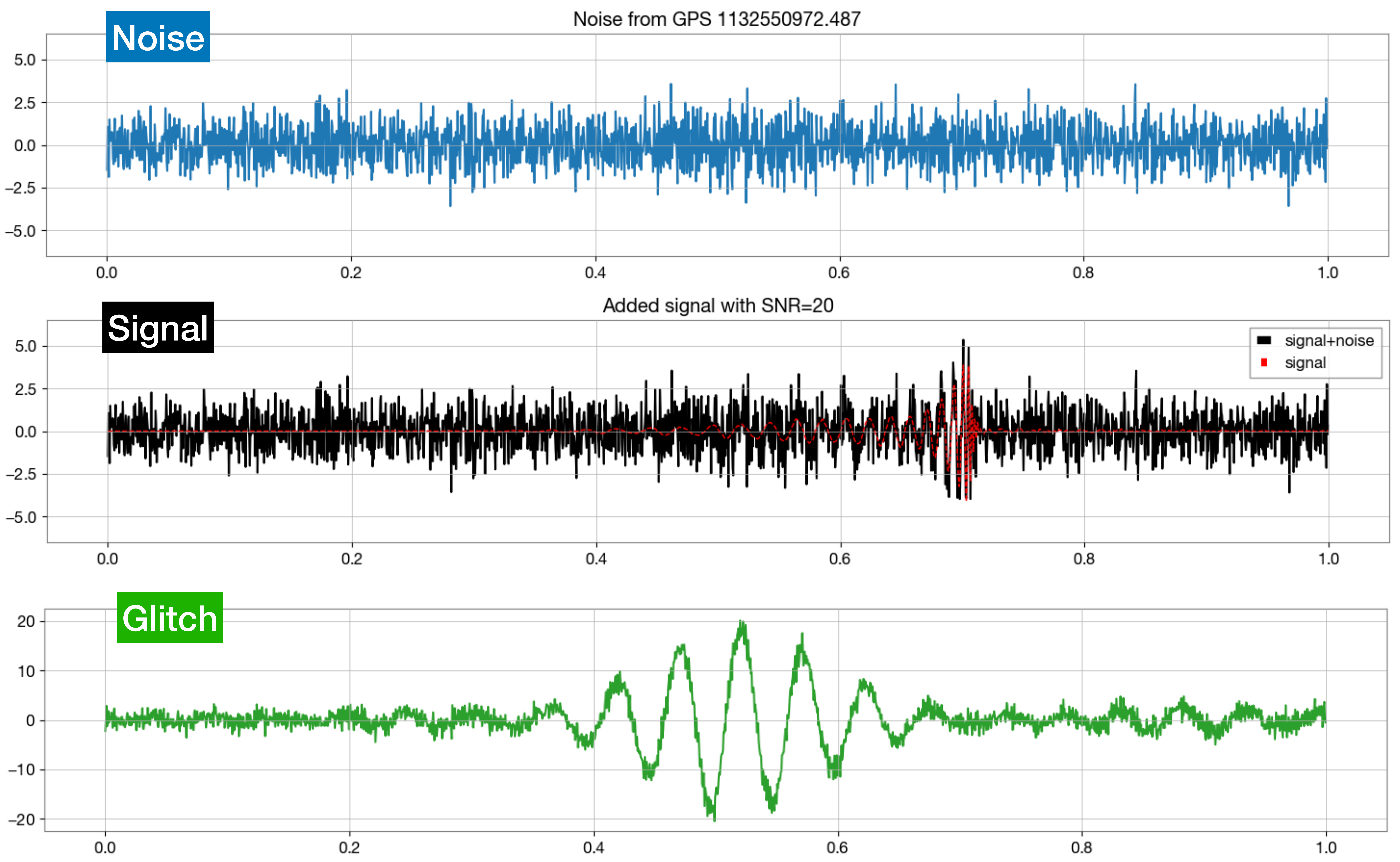
Training data: 3 classes

Segments of glitches and “almost Gaussian” noise data samples from the one month of LIGO O1 run (downsampled to 2048 Hz), whitened by the amplitude spectral density of the noise.

Real detector noise from real data when nor glitches nor signals nor injections are present

Real detector noise (selected as noise class) + BBH injections

Data containing glitches (glitches inferred from 2+ detector periods with gravity spy and cWB)



NN architectures

- CNN : Convolutional Neural Network
  - ✓ Similar choice to previous works
- TCN : Temporal Convolutional Network
- IT : Inception Time
  - Modern architectures based on CNN but conceived for time series classification
  - Applied to this problem for the first time

Input time series data

↓

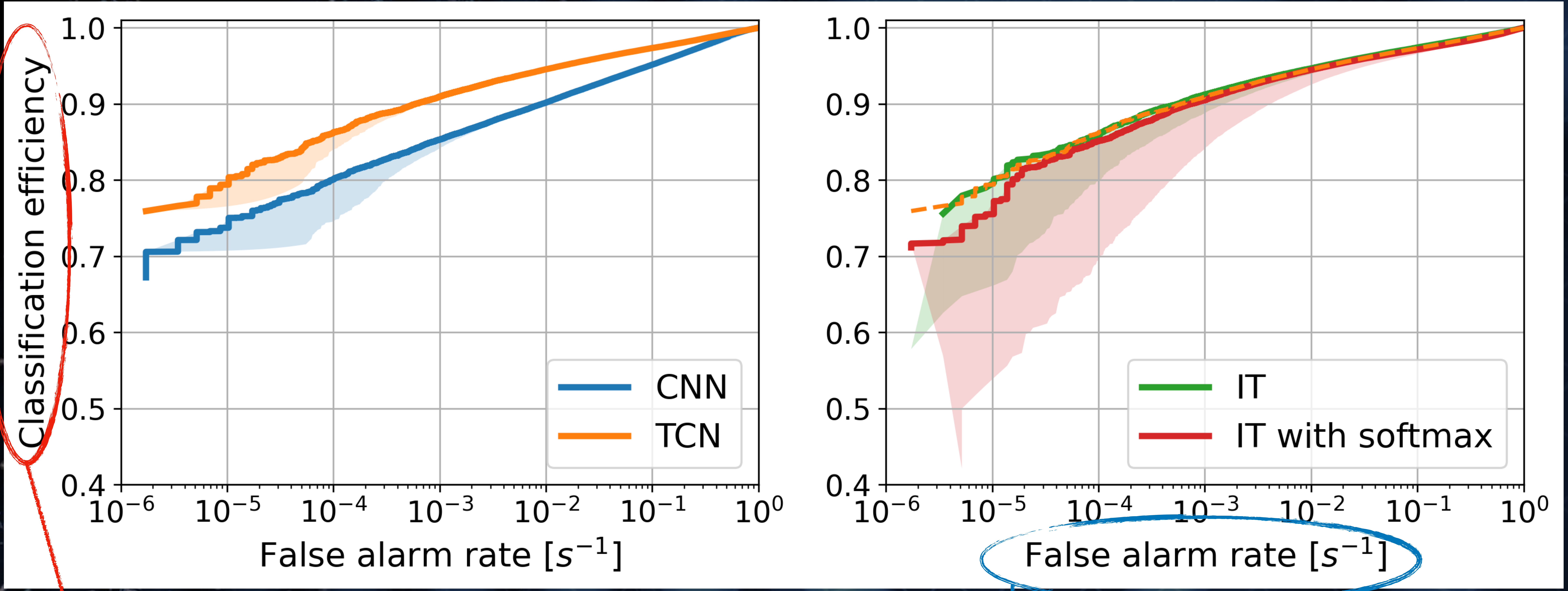
Neural network

↓

Probability for each of the three classes

- After a rough optimisation of the hyperparameters of each model, we fixed them and trained and tested the same model 10 times, choosing the model with the highest ROC (see next slides)

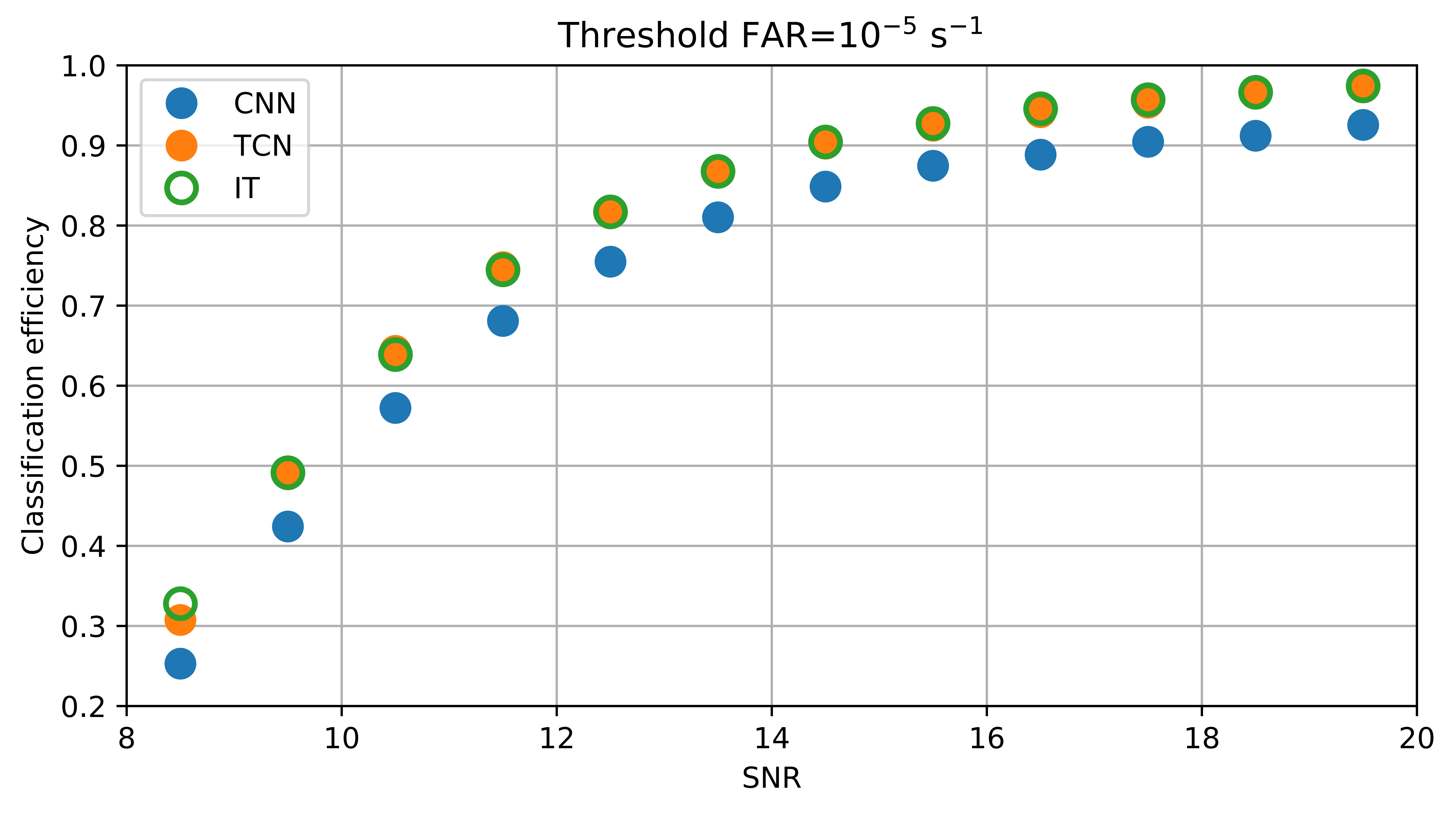
ROC curves



$$\frac{\#\ \textit{signal samples with } P_s \textit{ above some threshold}}{\textit{Tot signal samples}}$$

$$\frac{\#\ \textit{noise} + \textit{glitch samples with } P_s \textit{ above some threshold}}{\textit{Tot duration}[s]\ \textit{noise} + \textit{glitch samples}}$$

Classification efficiency vs SNR for fixed FAR



Only the best model out of the 10 repetitions considered for each architecture

- TCN and IT perform similarly and outperform CNN
- Efficiency better than 0.5 for SNR>9 at this level of FAR
  - (1 alarm per 10^5 s = 0.864 alarms per day)

Trigger selection cut

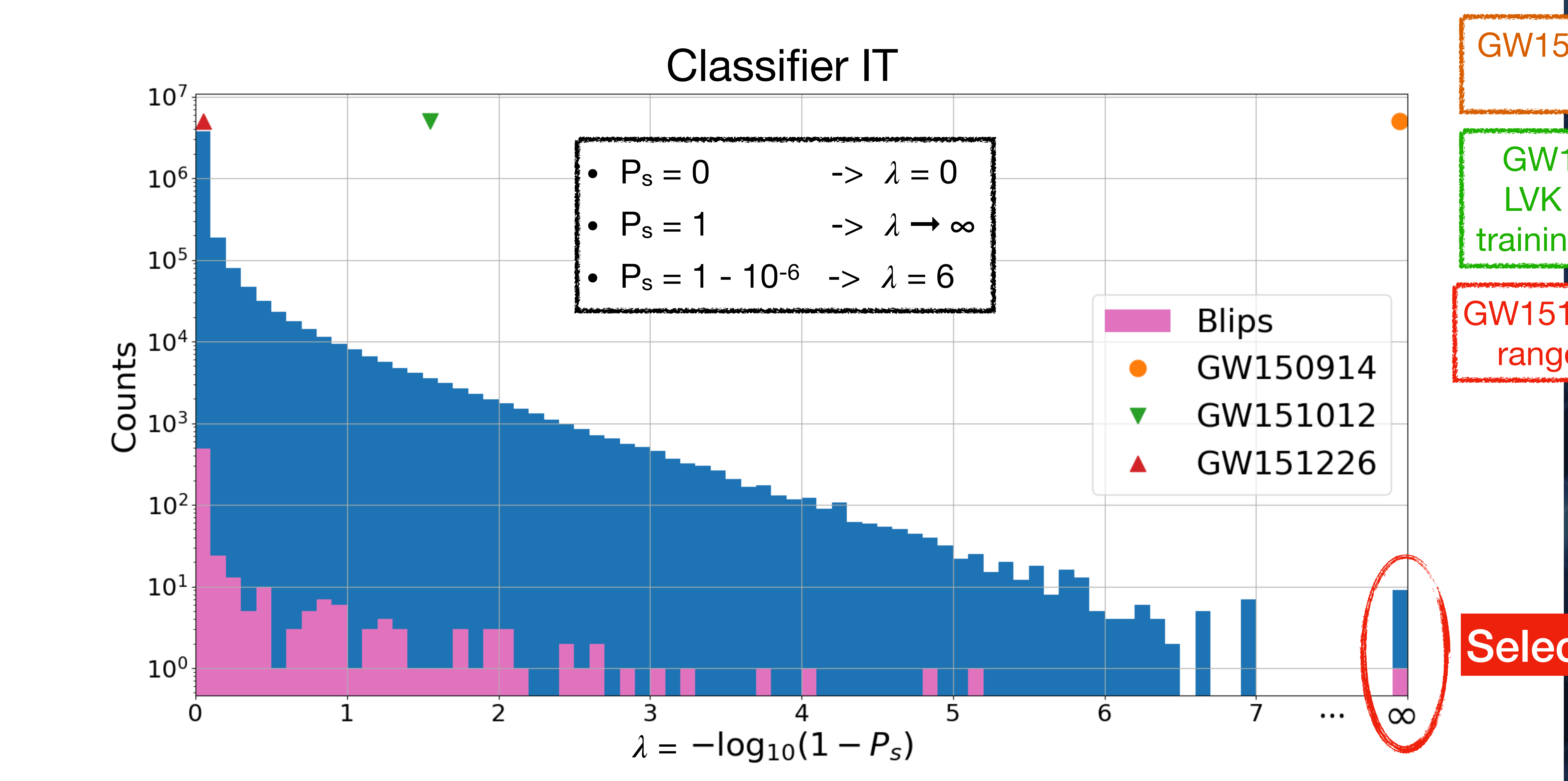
- We focus on the stricter cut that we can consider: **P_s=1** at machine precision (single-precision floating-point format)
- With this cut we have:

	CNN	TCN	IT
Noise+glitch samples with P_s=1	0	1	2
Equivalent FAR [s^{-1}]	$< 1.7 \times 10^{-6}$	1.7×10^{-6}	3.4×10^{-6}
Equivalent FAR in days	< 1/(7 days)	1/(7 days)	1/(3 days)
Signal classification efficiency	65%	76%	76%

- The FAR level reached is compatible with our initial goal: 2 false alarms per day => FAR = 2.3×10^{-5} s^{-1}

Analysis of the remaining 3 months of O1

- We applied the 3 networks to the remaining 3 months of L1 in O1 excluding the 1 month period already used for training and testing and know injections
- Periods around known GW detections have been examined separately



GW150914 identified with $P_s = 1$ by all networks

GW151012 was detected by LVK in L1 with a SNR~6 (our training set has a minimum of 8)

GW151226 has masses not in the range used in our training set

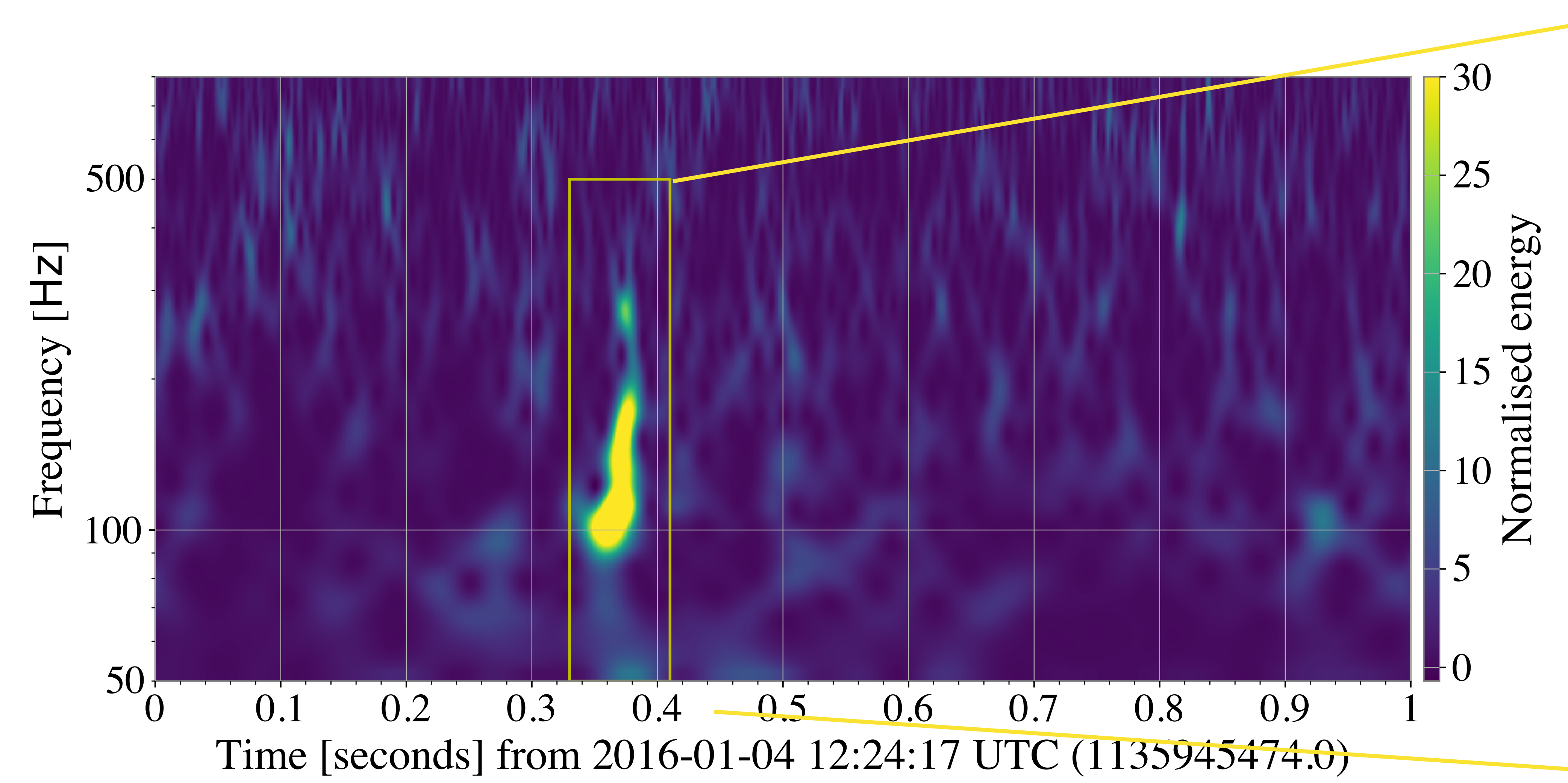
Selected triggers

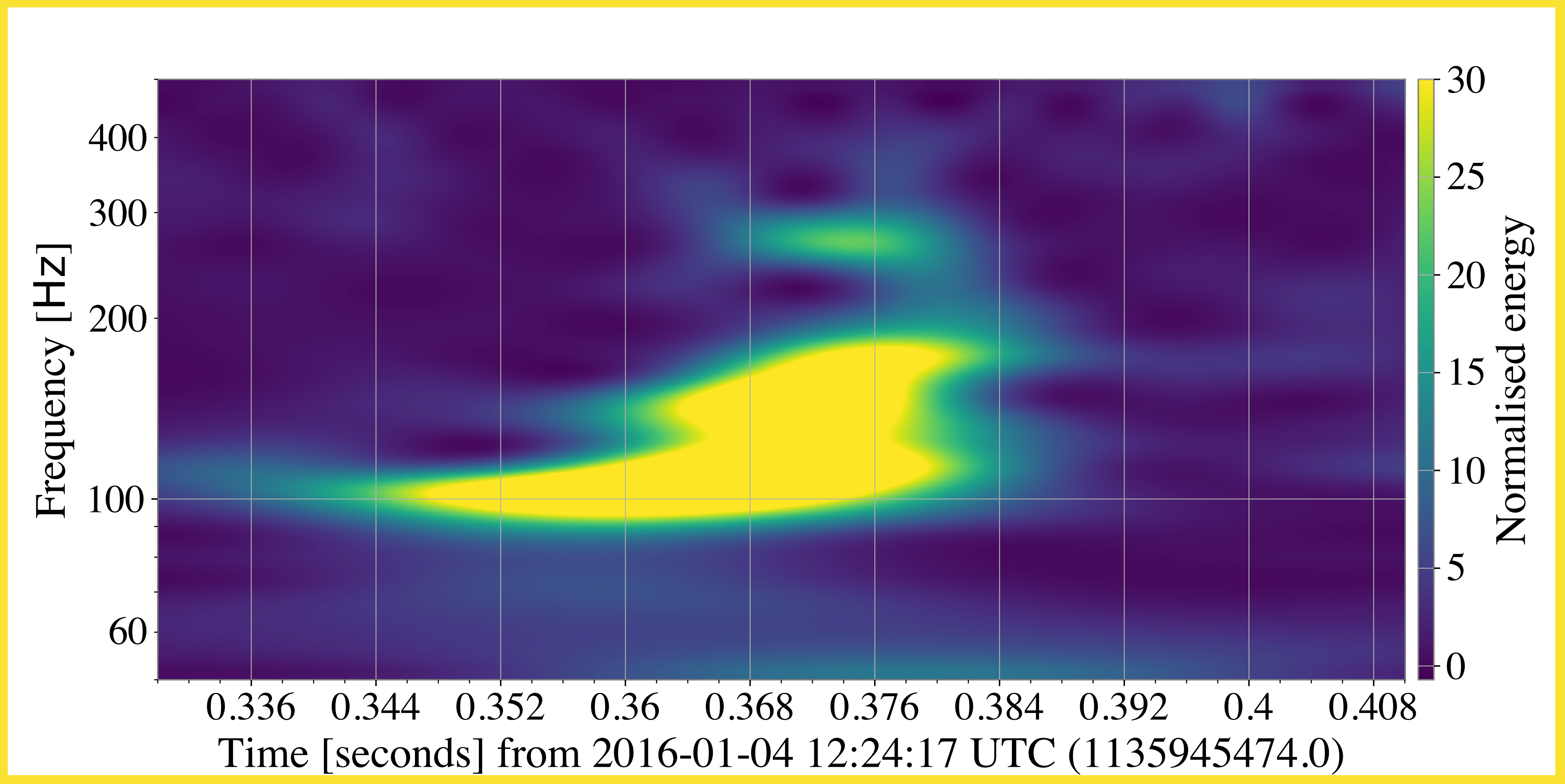
Triggers found in the remaining 3 months of O1

- Selection cut: P_S=1

	CNN	TCN	IT
Samples with P_S=1 in single-det time	2	14	2
Samples with P_S=1 in double-det time	2	91*	7

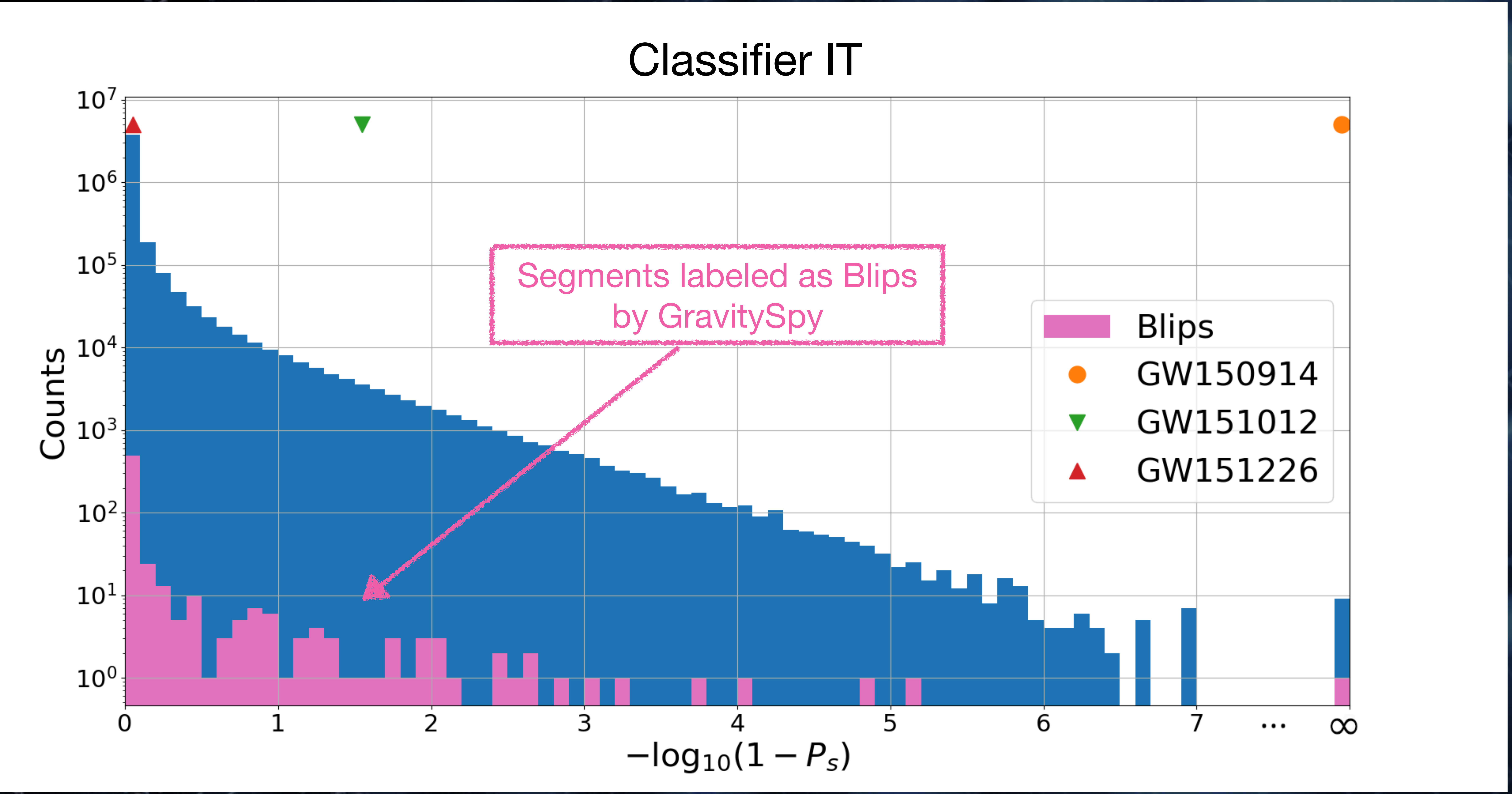
- Only one event common to the three analyses: L1-only at **GPS=1135945474.0 (2016-01-04 12:24:17 UTC)**

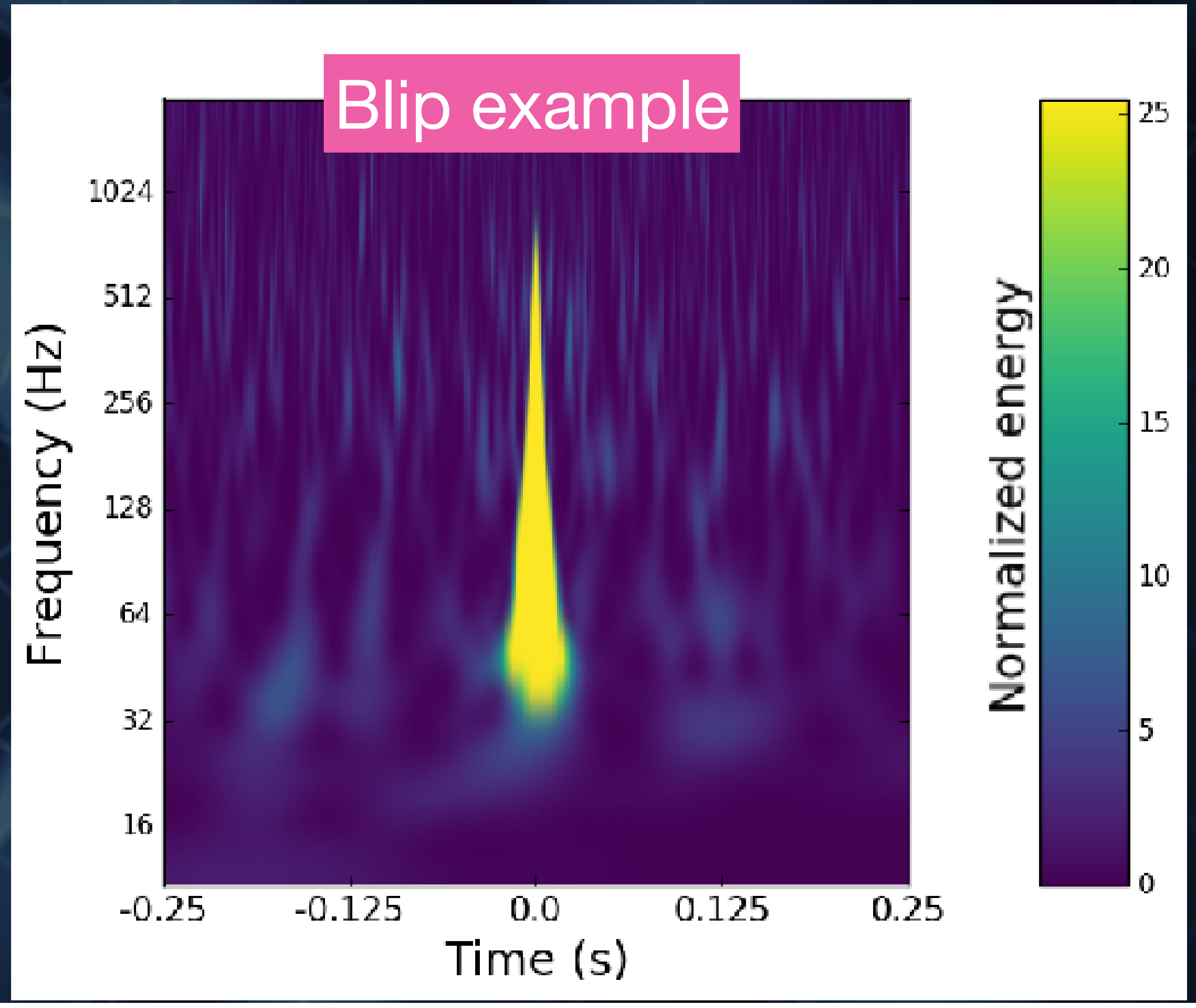


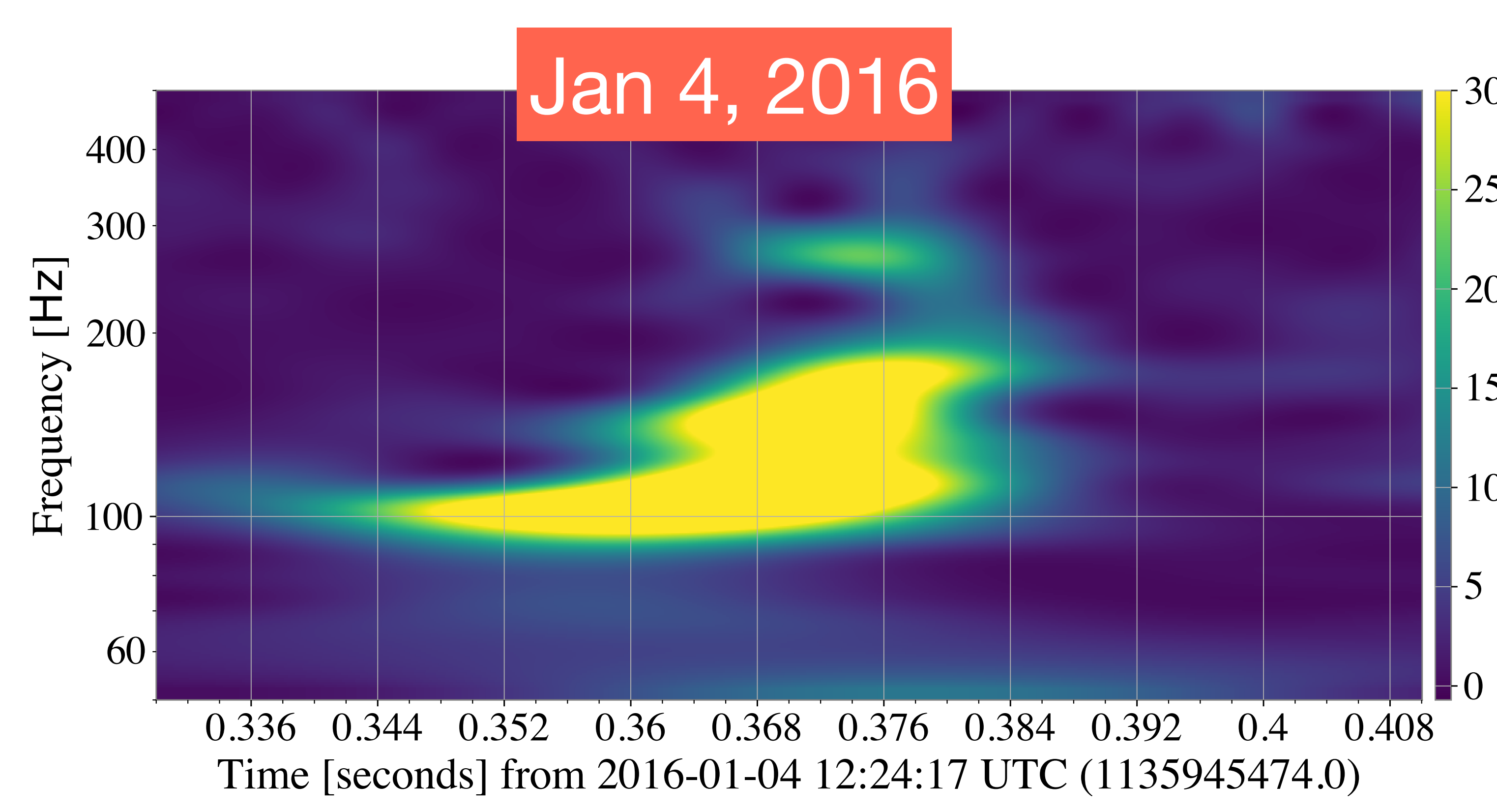


Is it a Blip?

- Gravity Spy finds a Blip at 1135945474.373
- In general the population of Blips compatible with background: Jan 4 outlier for this population

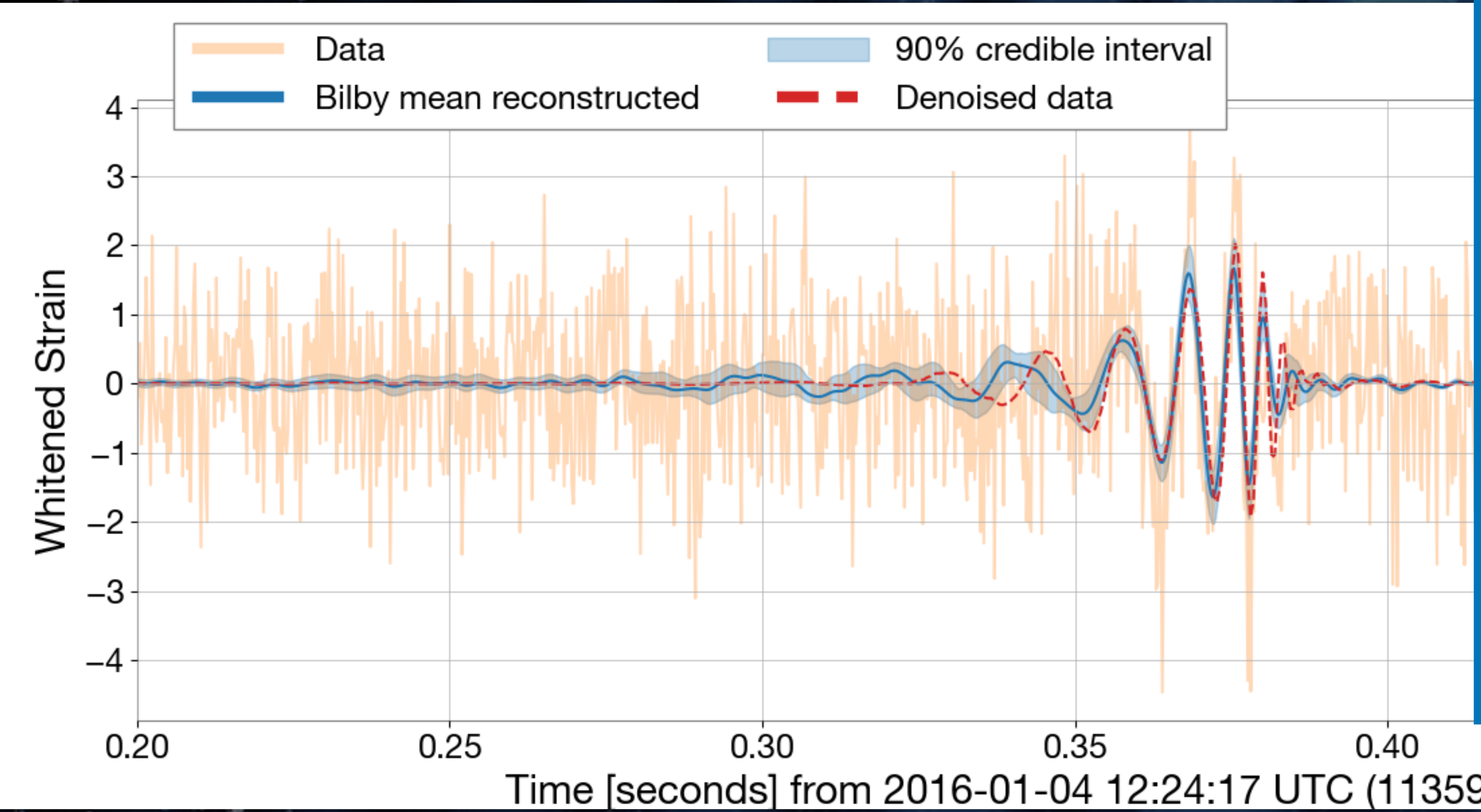




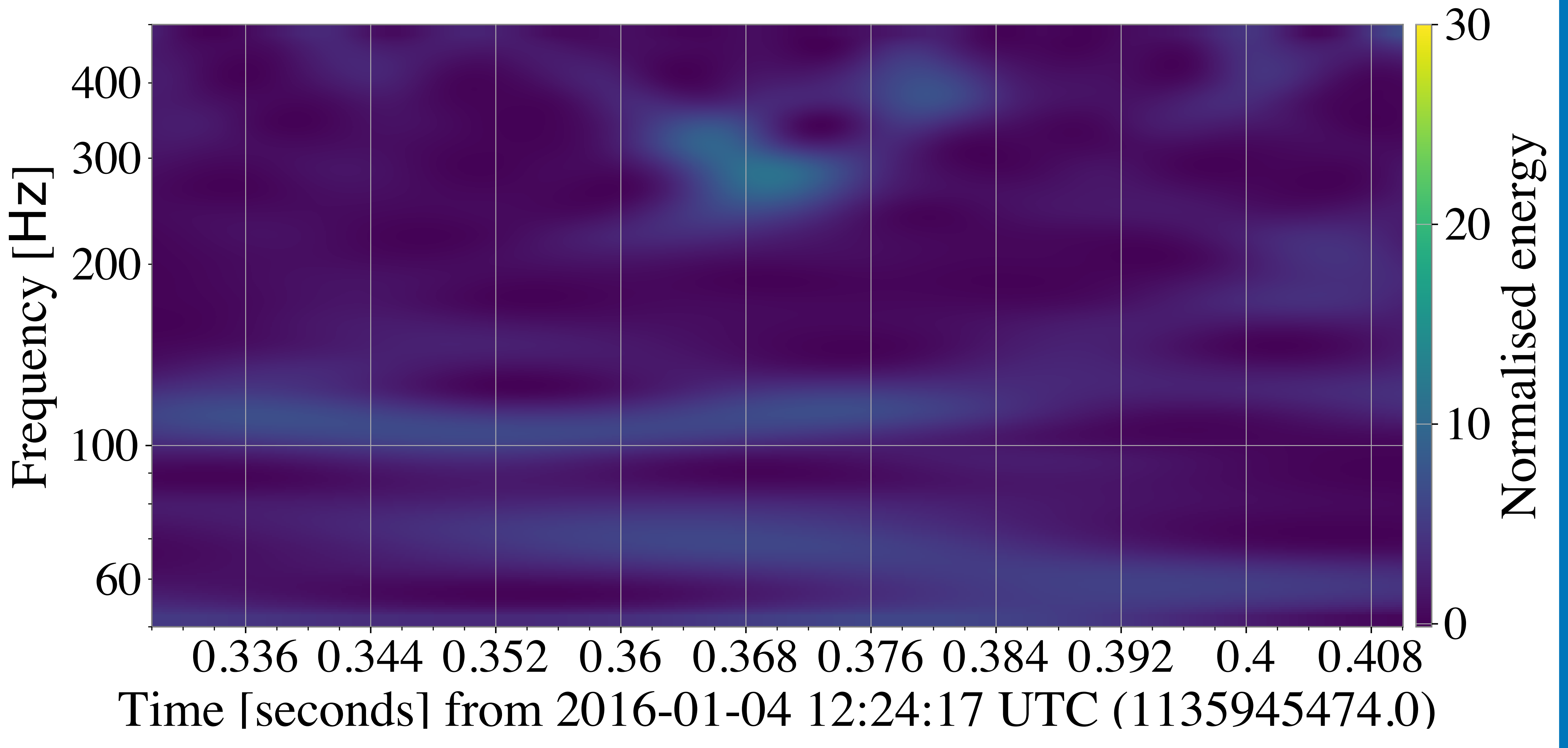


Has it an astrophysical origin?

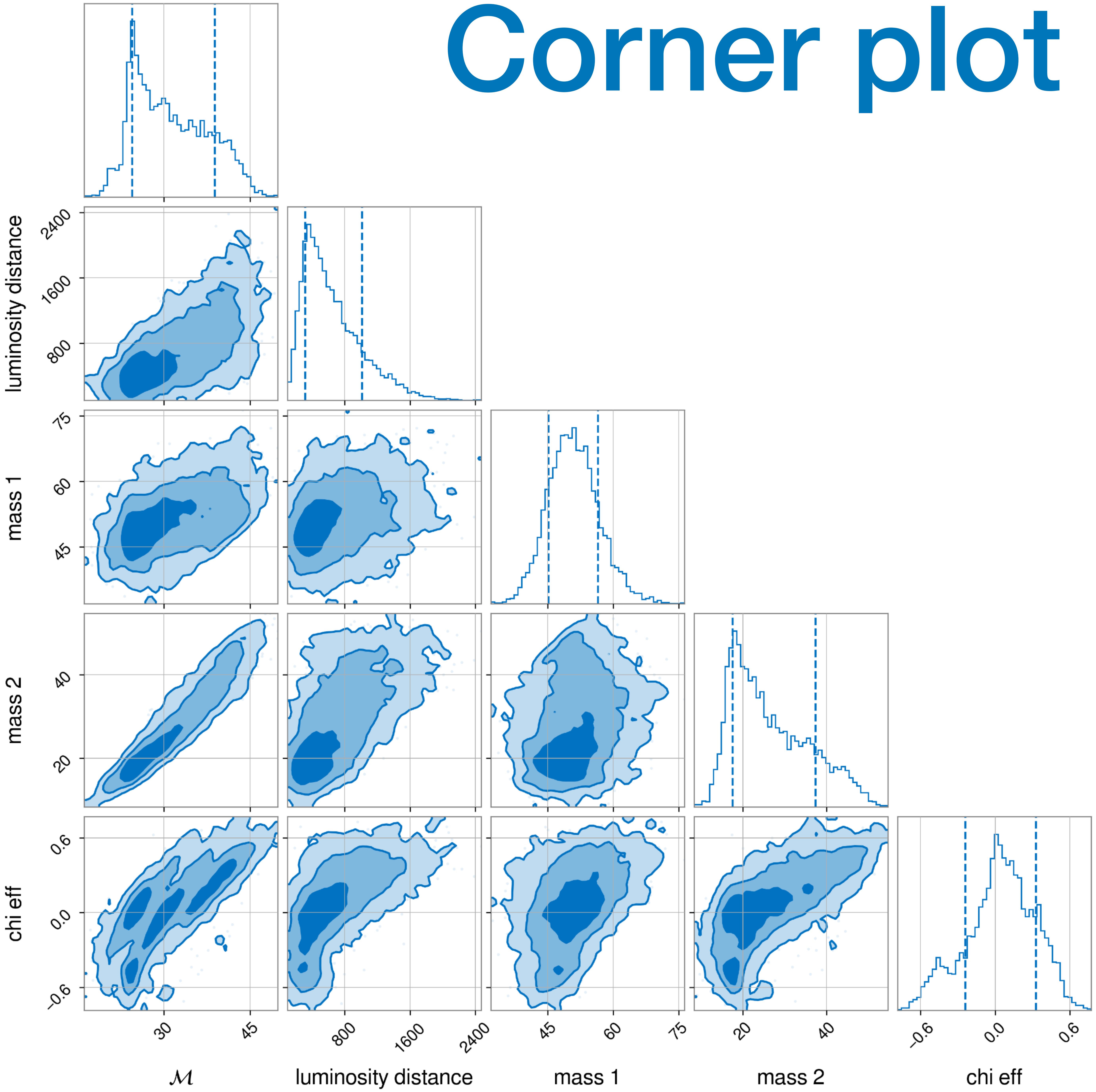
- Checks that the transient signal is compatible with a GW waveform model
  - ✓ Bayesian parameter estimation: Bilby
  - ✓ Independent check: denoising convolutional neural network by Bacon et al 2023 Mach. Learn.: Sci. Technol. 4 035024



Residual after subtraction of Bilby waveform



Corner plot



$$GPS = 1135945474.373^{+0.076}_{-0.07}$$

$$SNR = 11.34^{+1.8}_{-1.6}$$

$$\mathcal{M} = 30.18^{+12.3}_{-7.3} M_\odot$$

$$m_1 = 50.7^{+10.4}_{-8.9}\ M_\odot$$

$$m_2 = 24.4^{+20.2}_{-9.3}\ M_\odot$$

$$\chi_{\mathrm{eff}} = 0.06^{+0.4}_{-0.5}$$

$$d_L = 564^{+812}_{-338}\ \mathrm{Mpc}$$

Consistent with BBH population observed so far

Conclusion

- Architectures specifically designed for time-series classification, such as IT or TCN, outperform the standard CNN typically used so far
- 1 month of O1 L1 data used for training and testing: obtain reasonable noise rejection and detection efficiencies on single-detector data
  - ✓ Labeled dataset available on zenodo: https://zenodo.org/records/11093596
- Application of the models on the remaining 3 months of O1 L1 data
- All the classifiers independently detect on January 4, 2016
  - ✓ Possible astrophysical origin investigated and looks plausible
  - ✓ In the past other papers have investigated this event (Alexander H. Nitz et al 2020 ApJ 897 169)
- Currently working on O1 data from H1 + p-astro calculation

Acknowledgment

*This work is **(partially)** supported by ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU*

This research has made use of data or software obtained from the Gravitational Wave Open Science Center (gwosc.org), a service of the LIGO Scientific Collaboration, the Virgo Collaboration, and KAGRA. This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation, as well as the Science and Technology Facilities Council (STFC) of the United Kingdom, the Max-Planck-Society (MPS), and the State of Niedersachsen/Germany for support of the construction of Advanced LIGO and construction and operation of the GEO600 detector. Additional support for Advanced LIGO was provided by the Australian Research Council. Virgo is funded, through the European Gravitational Observatory (EGO), by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale di Fisica Nucleare (INFN) and the Dutch Nikhef, with contributions by institutions from Belgium, Germany, Greece, Hungary, Ireland, Japan, Monaco, Poland, Portugal, Spain. KAGRA is supported by Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan Society for the Promotion of Science (JSPS) in Japan; National Research Foundation (NRF) and Ministry of Science and ICT (MSIT) in Korea; Academia Sinica (AS) and National Science and Technology Council (NSTC) in Taiwan.

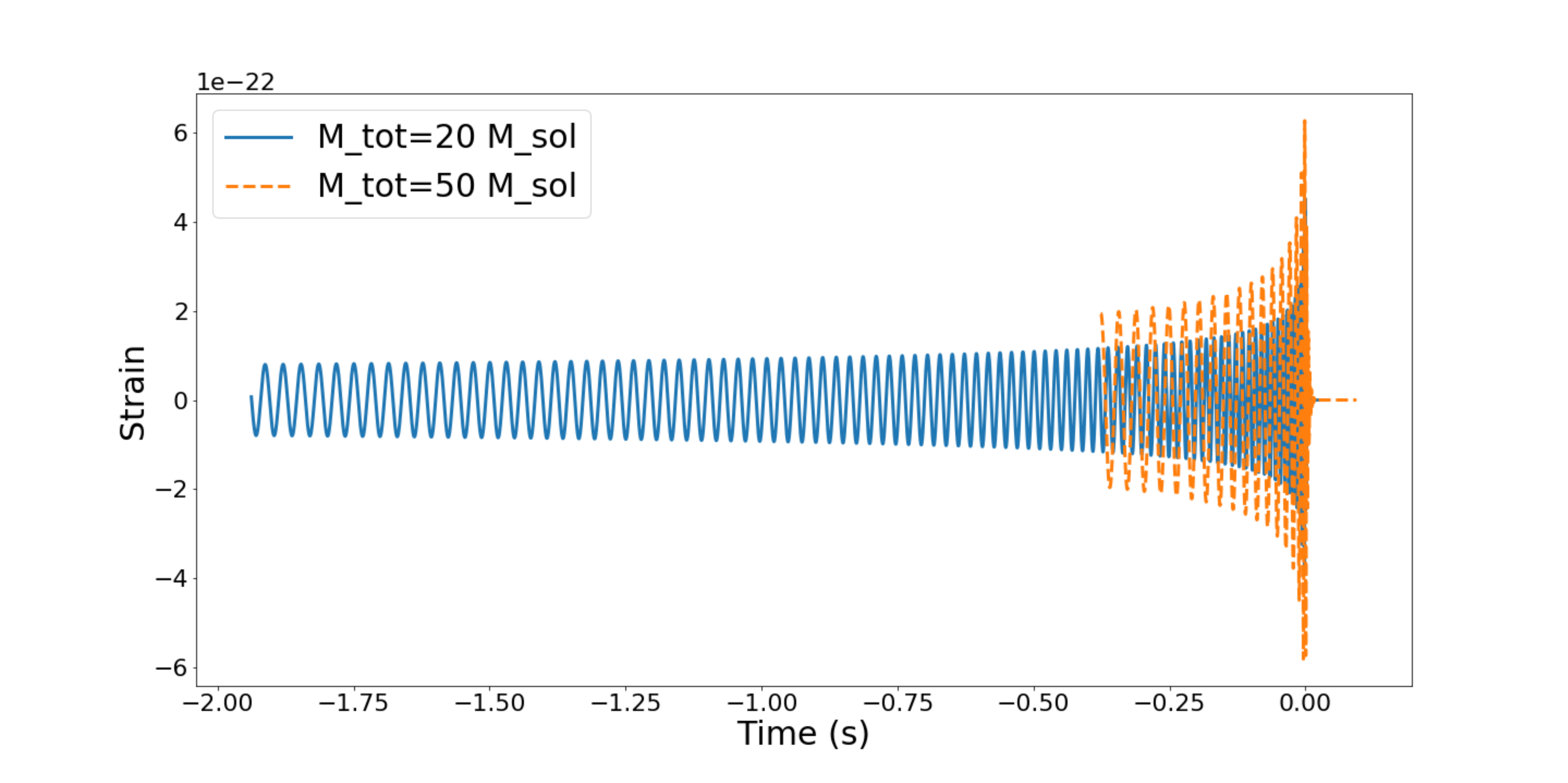
Backup slides

Training and testing datasets

- 1 month of L1 data without know GW detections (between Nov 25, 2015 and Dec 25, 2015)
- Segments of fixed duration: **1 second**
- **Bandpass filter [20,1000] Hz**
- **No superposition** between segments
- Glitch **position random** in the segment (if short duration, fully contained) or tailing over multiple segments if duration > 1 s
- Samples for training:
  - Noise: 2.5e5
  - Signal: 2.5e5
  - Glitch: 0.7e5
- Samples for testing:
  - Noise: 5e5
  - Signal: 5e5
  - Glitch: 0.8e5

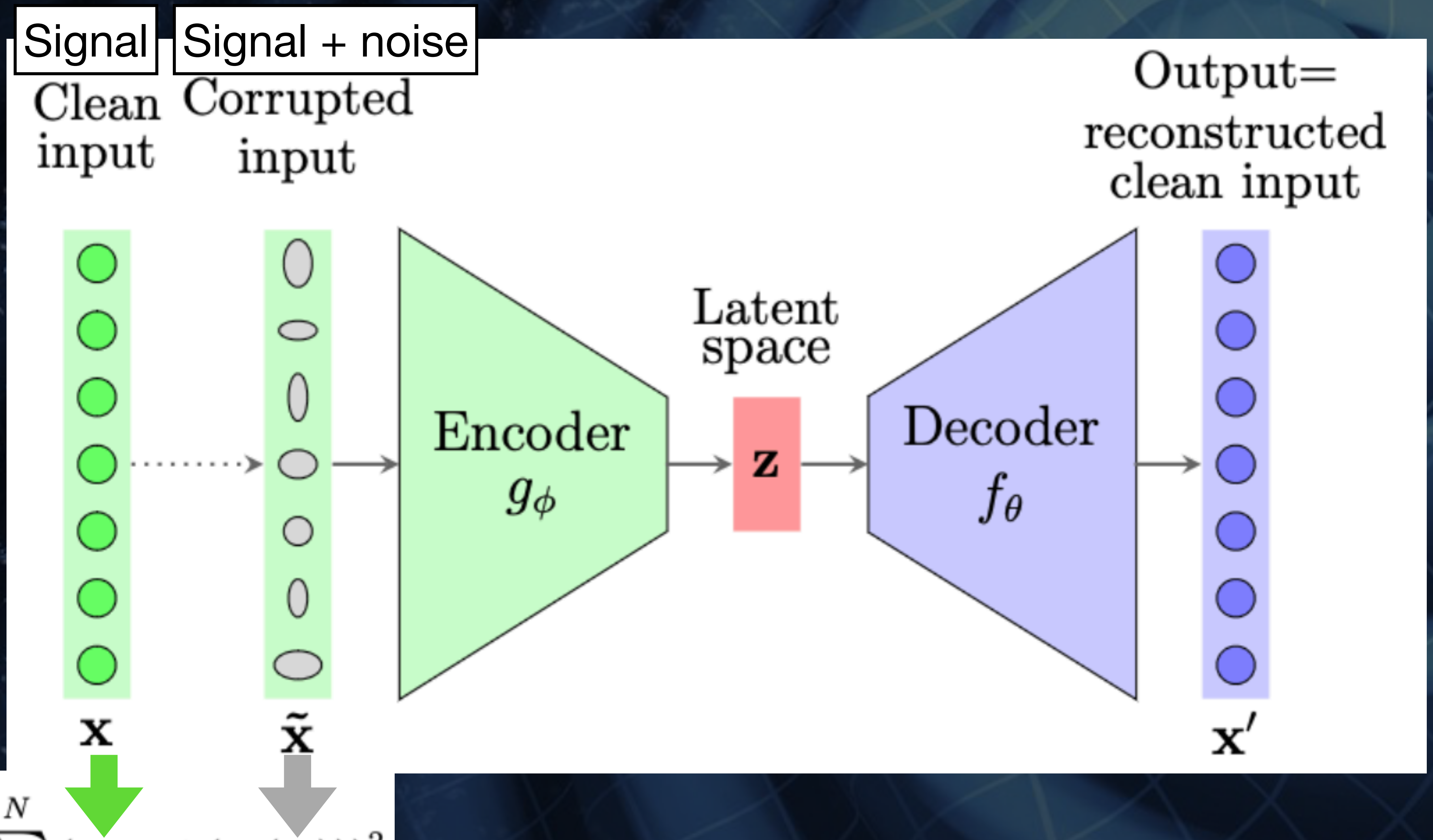
Signal injection:

- **Position random** in the segment but almost fully contained
- Type pf signal: (BBH, waveform model SEOBNRv4)
  - **m1,m2** $\in$ **(10,50) M⊙ & m1+m2** $\in$ **(33,60) M⊙**
  - **SNR** $\in$ **(8,20)**



Has it an astrophysical origin?

- Checks that the transient signal is compatible with a GW waveform model
  - ✓ Bayesian parameter estimation: Bilby
  - ✓ Independent check: denoising convolutional neural network by Bacon et al 2023 Mach. Learn.: Sci. Technol. 4 035024



$$L_{DAE}(\theta, \phi) = \sum_{i=1}^{N} (x_i - f_\theta(g_\phi(\tilde{x}_i)))^2$$

Denoising: model that takes noisy signals and returns clean signals

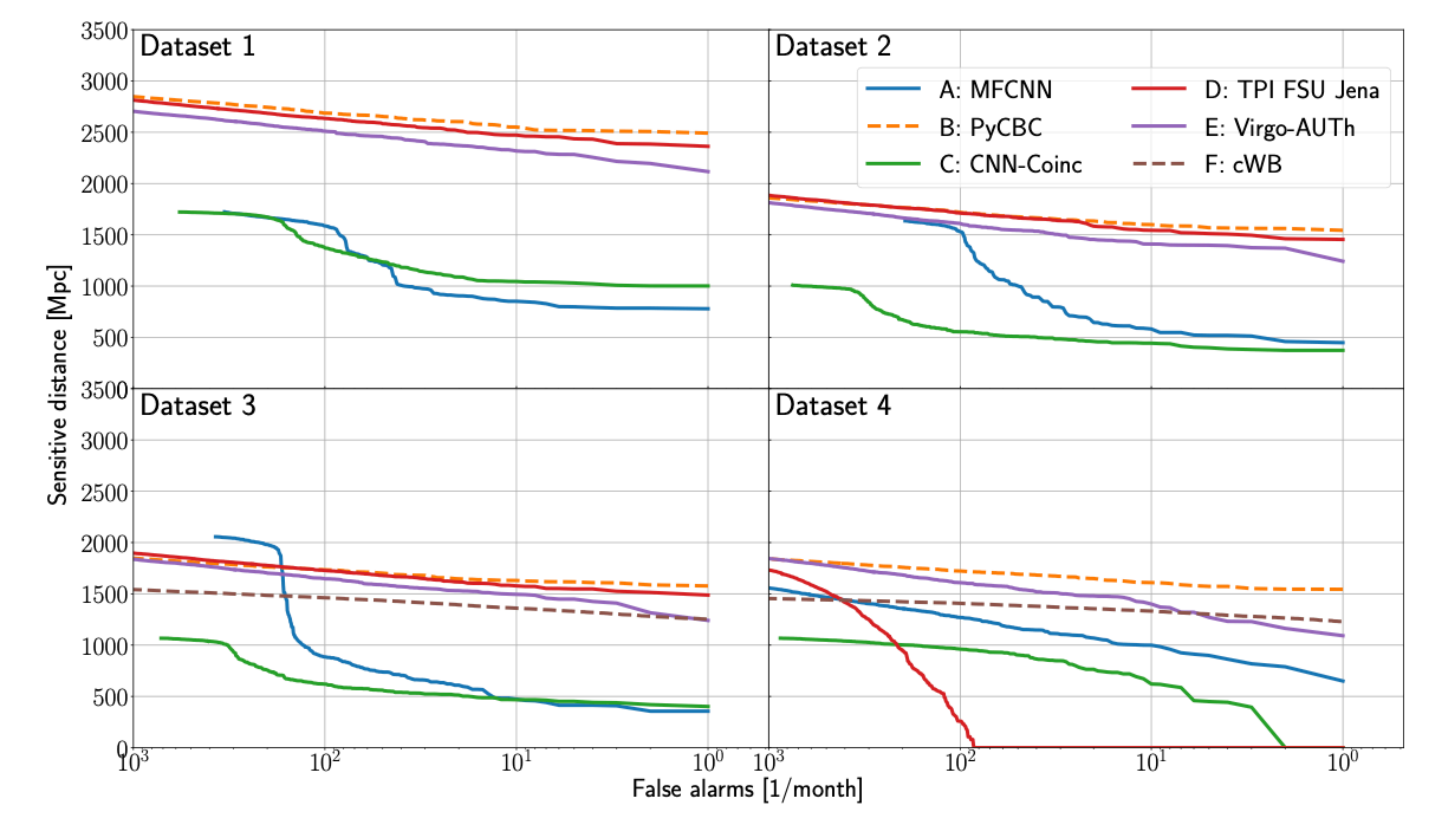
Enconder and decoder are CNNs

ML used for GW signal detection

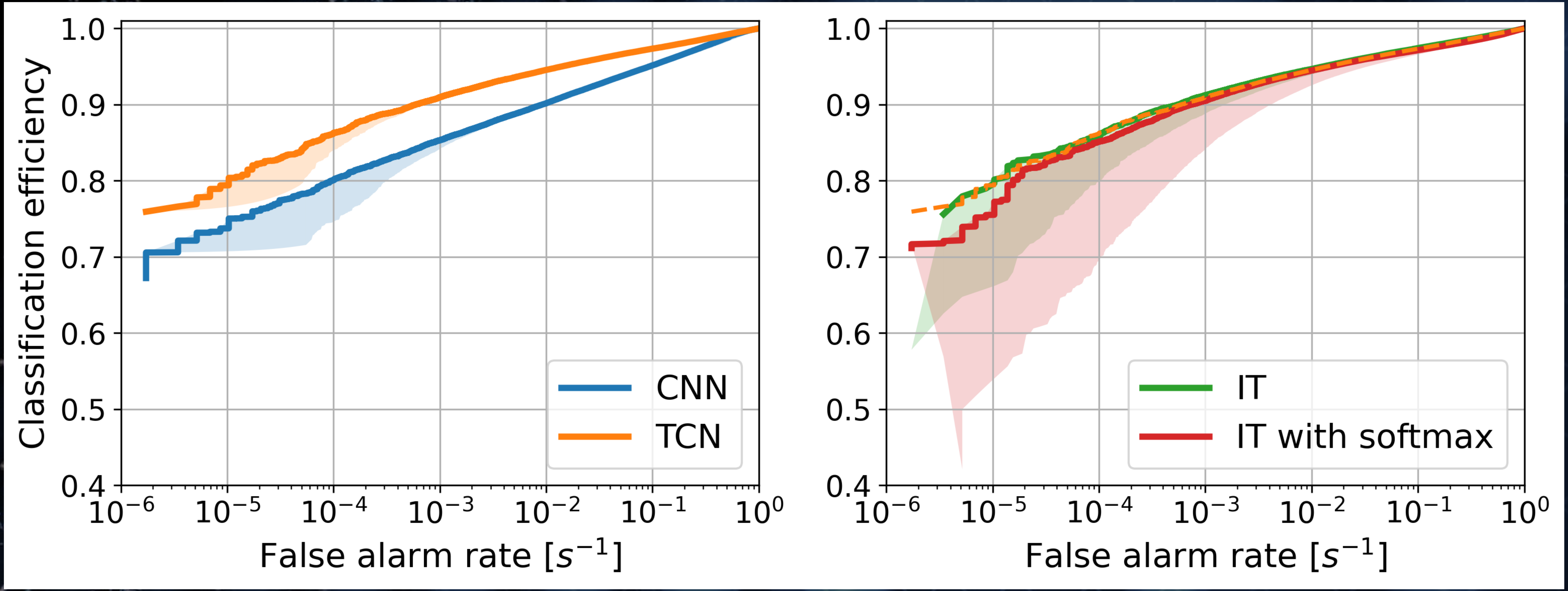
- **Lot of literature see e.g. this page: https://iphysresearch.github.io/Survey4GWML/#fn:174**

➡ Example: M. B. Schäfer et al. Phys. Rev. D 107 (2023) 023021

✓ Multi-detector search



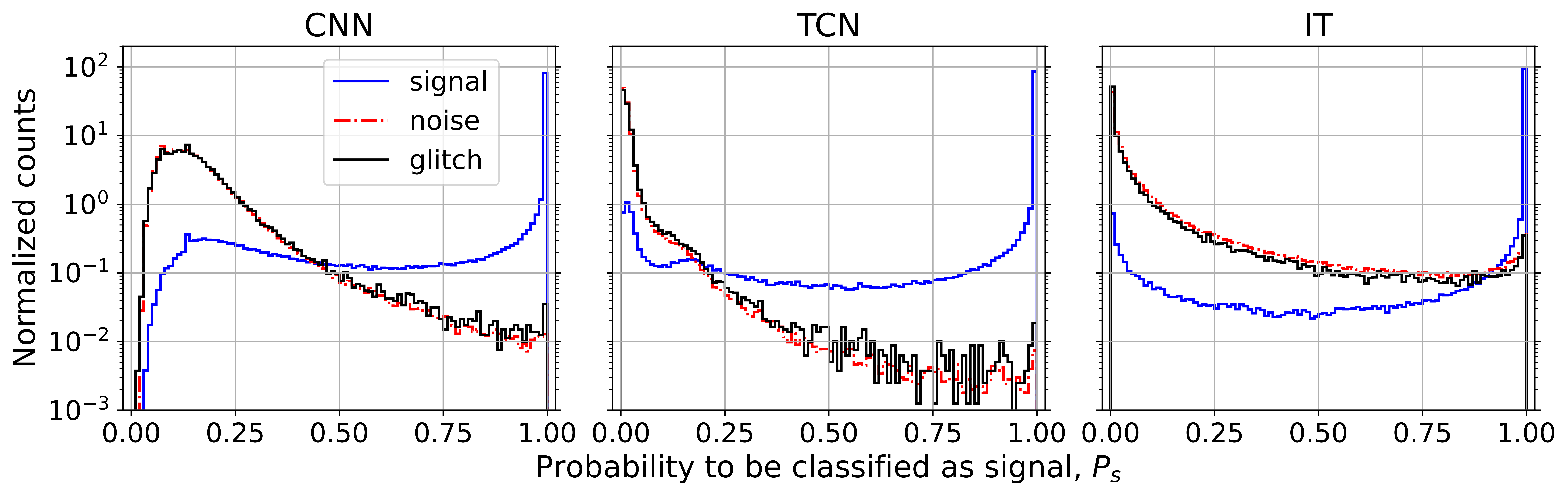
ROC curves



- Shaded area between the highest and the lowest ROC curves obtained for each model in the 10 repetitions of train and test
- "IT with softmax" refers to IT model with softmax activation function applied at the last fully connection layer during training.

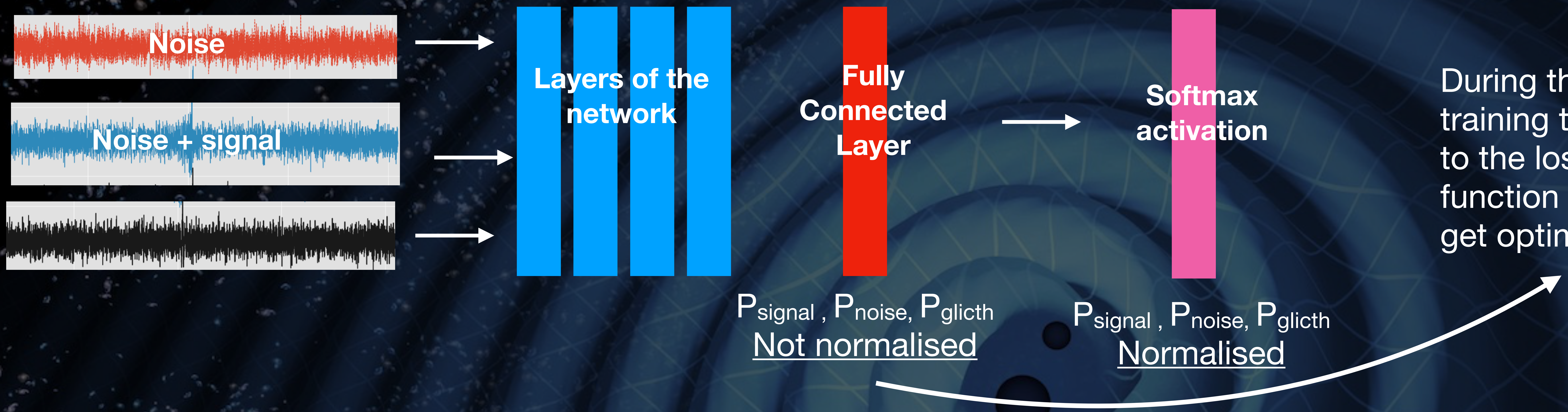
Probability to be classified as signal

Probability to be classified as signal can be used as test statistic



- Noise and glitch classes looks similar in all cases because in general the networks are not able to distinguish between glitch and noise (so they behave as only one class actually)
- We decided to focus on the signal identification and sum up noise + glitch

Softmax activation



During the training this goes to the loss function which get optimised

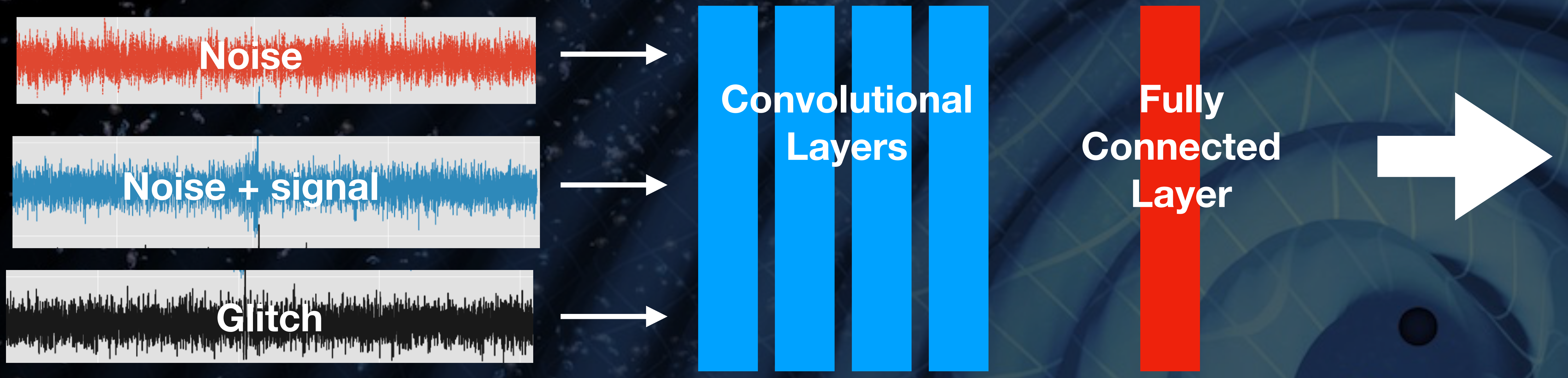
- We removed the use of the softmax activation step during the training, so that the loss function receives directly the output form the fully connected layer
- This was useful because often the membership probabilities in output of the softmax activation are close to one and their numerical precision can create problems and TCN and IT had an improvement when removing this activation
- However when all the training is done the final output of the last epoch needs the use of only one last softmax activation to get normalised membership probabilities

Single-precision floating-point format

- Single precision = significand precision: 24 bits (23 explicitly stored)
- The closest P_s can get to 1 (without being 1) is $P_s = 1 - 2^{-24}$
- When calculating lambda out of it one gets: $-\log_{10}(1 - P_s) = 7.22$

CNN used as starting point

- CNN used: small network with 4 convolution layers (with dropouts and pooling) used as classifier to distinguish the 3 classes: noise, noise+signal, glitches



Output: probability of belonging to each class

Layer #	1	2	3	4	5
Type	Conv	Conv	Conv	Conv	Dense
Filters	256	128	64	64	-
Kernel	16	8	8	4	-
Strides	4	2	2	1	-
Activation	relu	relu	relu	relu	softmax
Dropout	0.5	0.5	0.25	0.25	-
Max Pool	4	2	2	2	-

Optimiser: Adam

Temporal Convolutional Network

- Web page: https://github.com/philipperemy/keras-tcn
- Paper: https://arxiv.org/abs/1803.01271
- Easy to install: *pip install keras-tcn*

2017).) The distinguishing characteristics of TCNs are: 1) the convolutions in the architecture are causal, meaning that there is no information "leakage" from future to past; 2) the architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNN. Beyond this, we emphasize how to build very long effective history sizes (i.e., the ability for the networks to look very far into the past to make a prediction) using a combination of very deep networks (augmented with residual layers) and dilated convolutions.

Pay attention to the **receptive field** (you how far the model can see in terms of timesteps)

$$R_{field} = 1 + 2 \cdot (K_{size} - 1) \cdot N_{stack} \cdot \sum_i d_i$$

Arguments of the TCN

```
TCN(
    nb_filters=64,
    kernel_size=3,
    nb_stacks=1,
    dilations=(1, 2, 4, 8, 16, 32),
    padding='causal',
    use_skip_connections=True,
    dropout_rate=0.0,
    return_sequences=False,
    activation='relu',
    kernel_initializer='he_normal',
    use_batch_norm=False,
    use_layer_norm=False,
    use_weight_norm=False,
    **kwargs
)
```

Same number of filters and kernel size in all the layers

By default 6 layers

Results given here: nb_filters=32, kernel_size=16

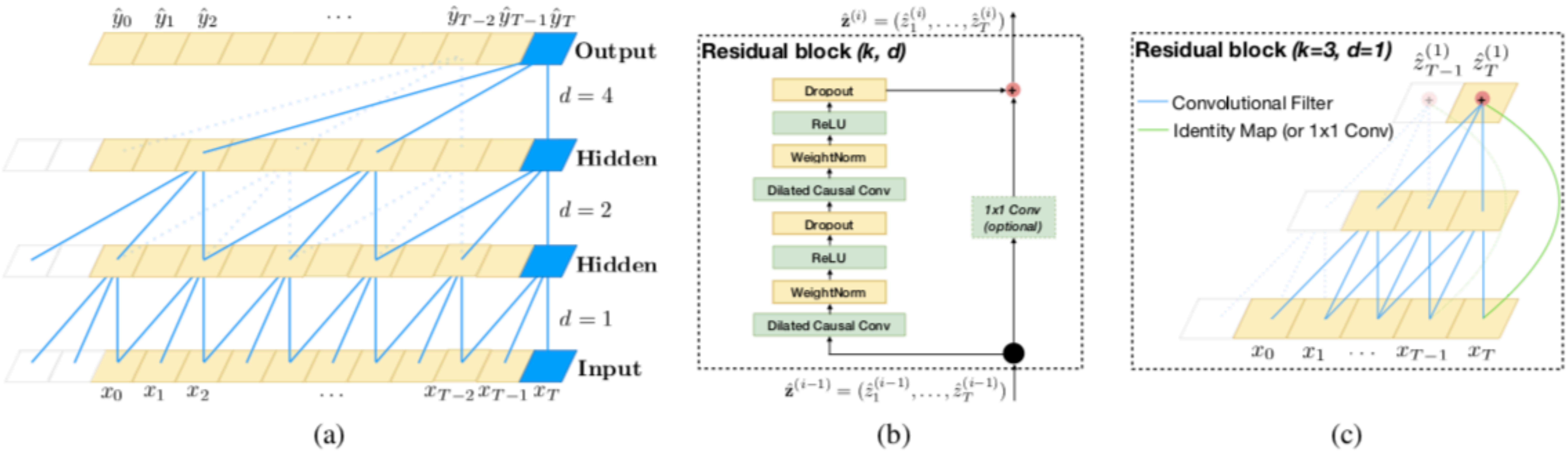
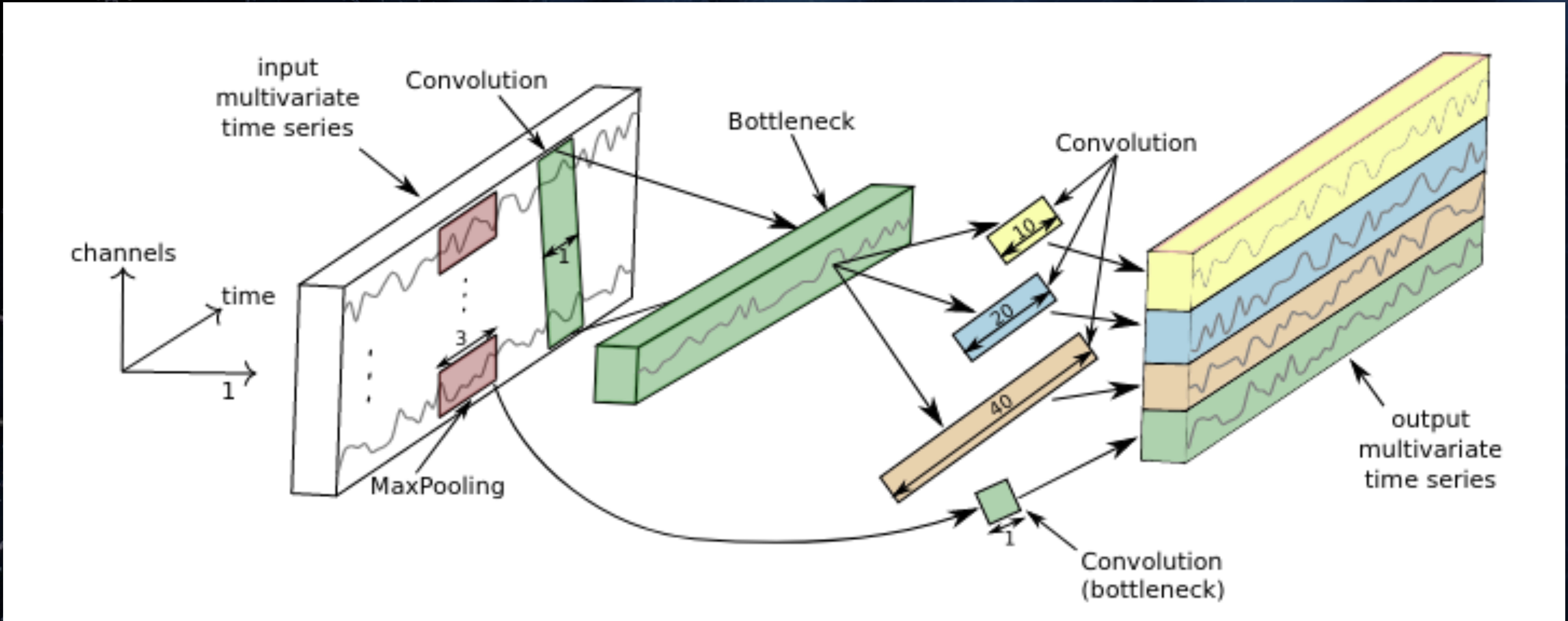


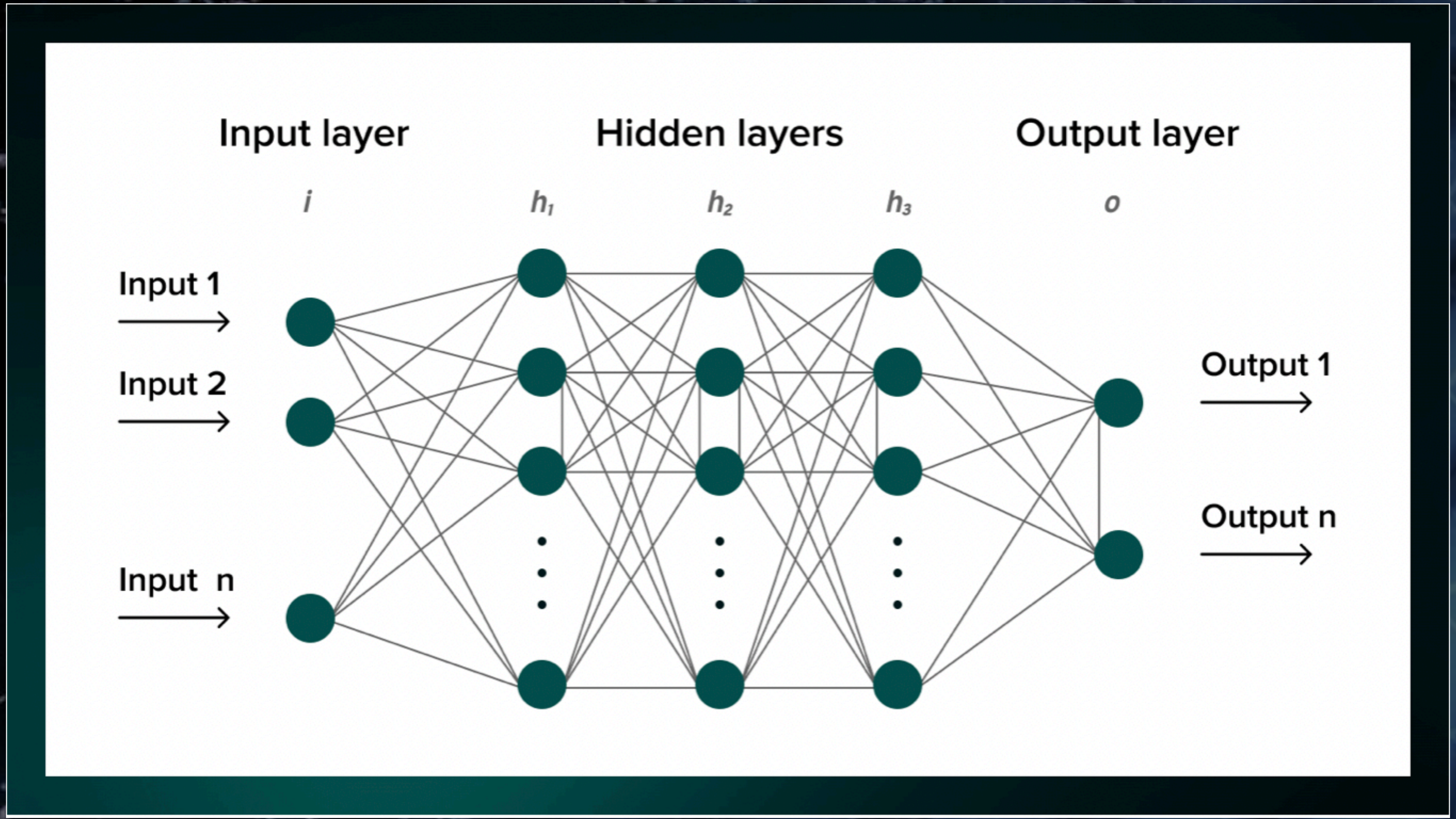
Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.

Inception time

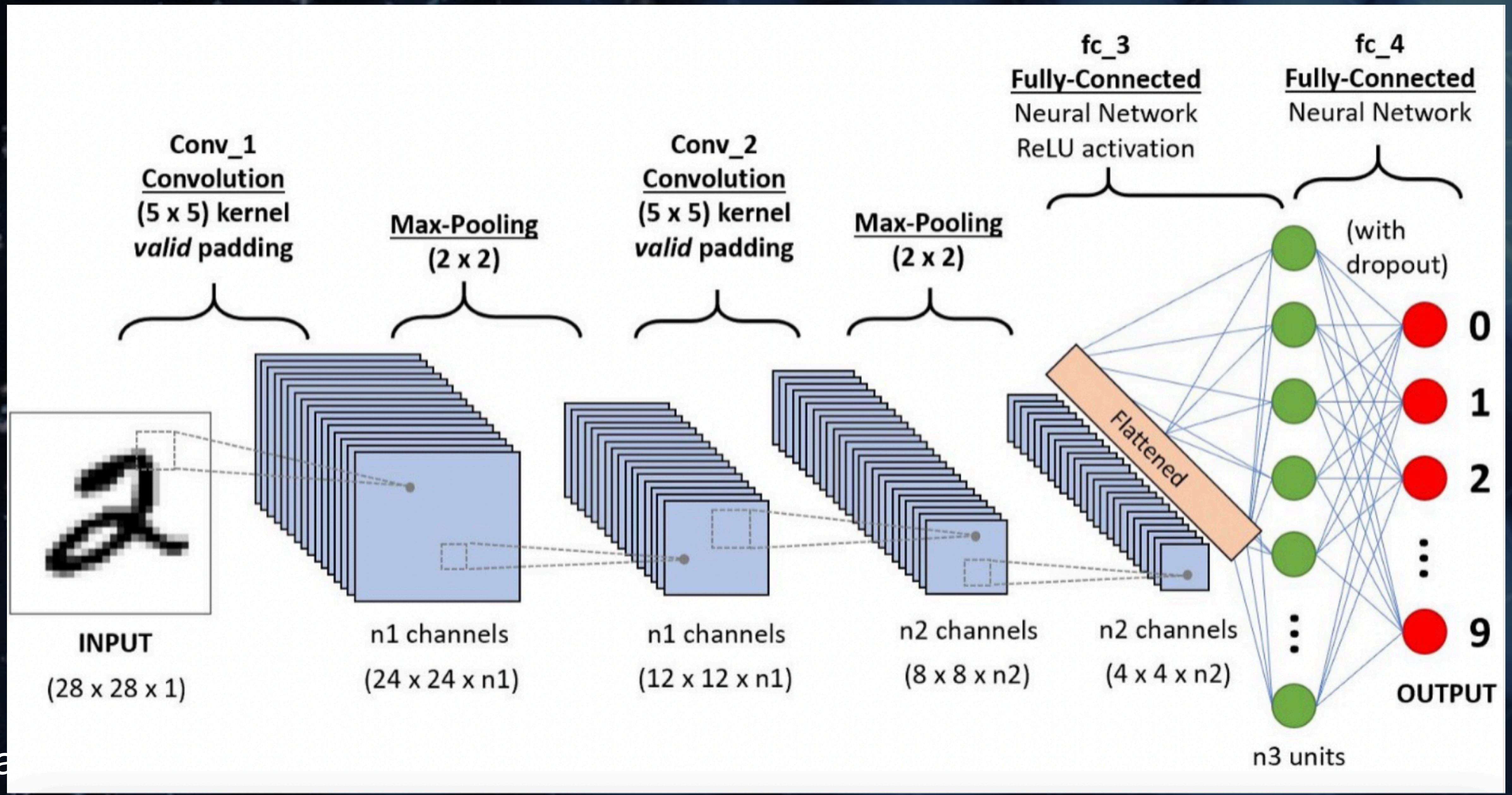


- https://arxiv.org/abs/1909.04939)

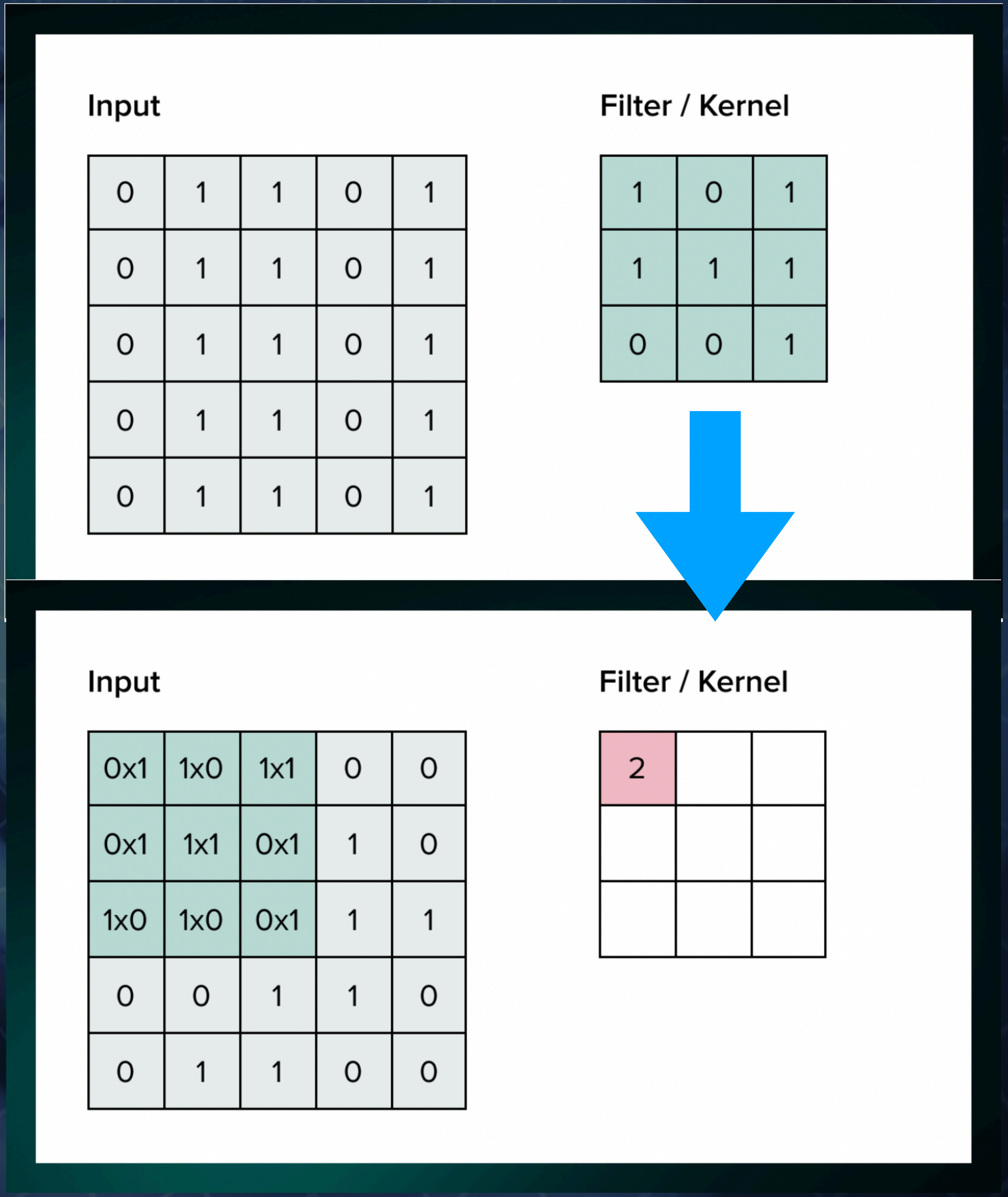
CNN



Input layer
Hidden layers
Output layer
i
h1
h2
h3
o
Input 1
Input 2
Input n
Output 1
Output n



Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with
dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT



Input
Filter / Kernel
0 1 1 0 1
0 1 1 0 1
0 1 1 0 1
0 1 1 0 1
0 1 1 0 1
1 0 1
1 1 1
0 0 1
Input
Filter / Kernel
0x1 1x0 1x1 0 0
0x1 1x1 0x1 1 0
1x0 1x0 0x1 1 1
0 0 1 1 0
0 1 1 0 0
2