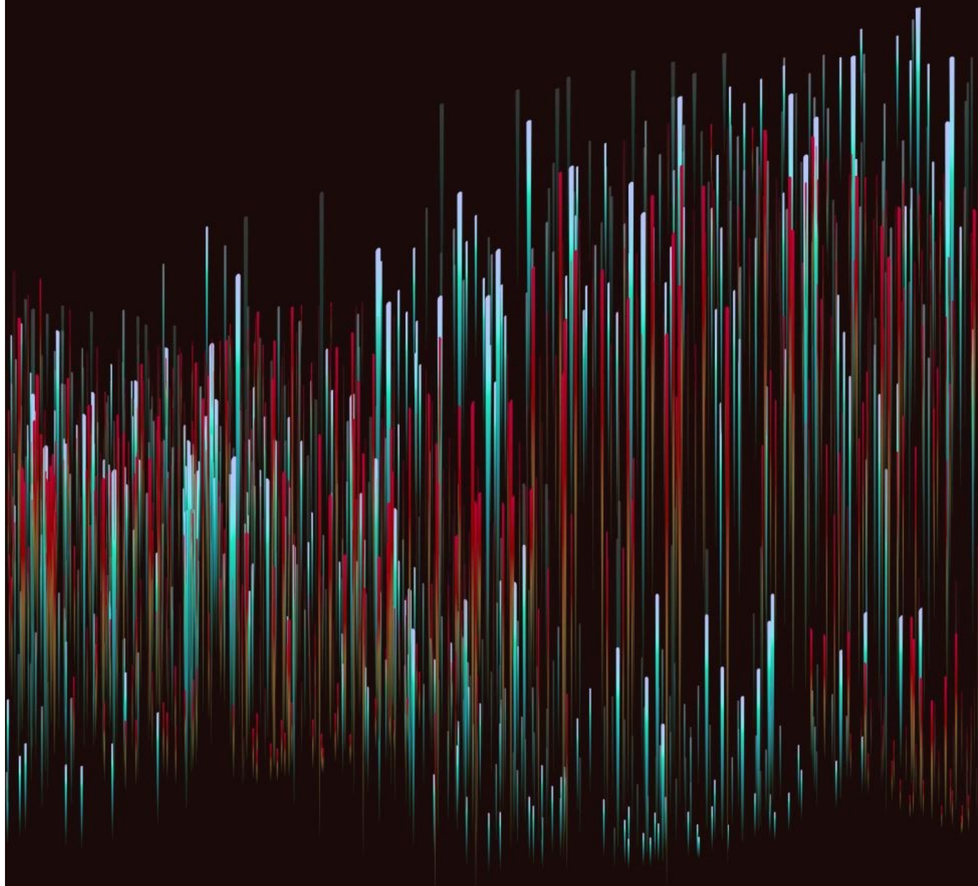
Glitch Generation

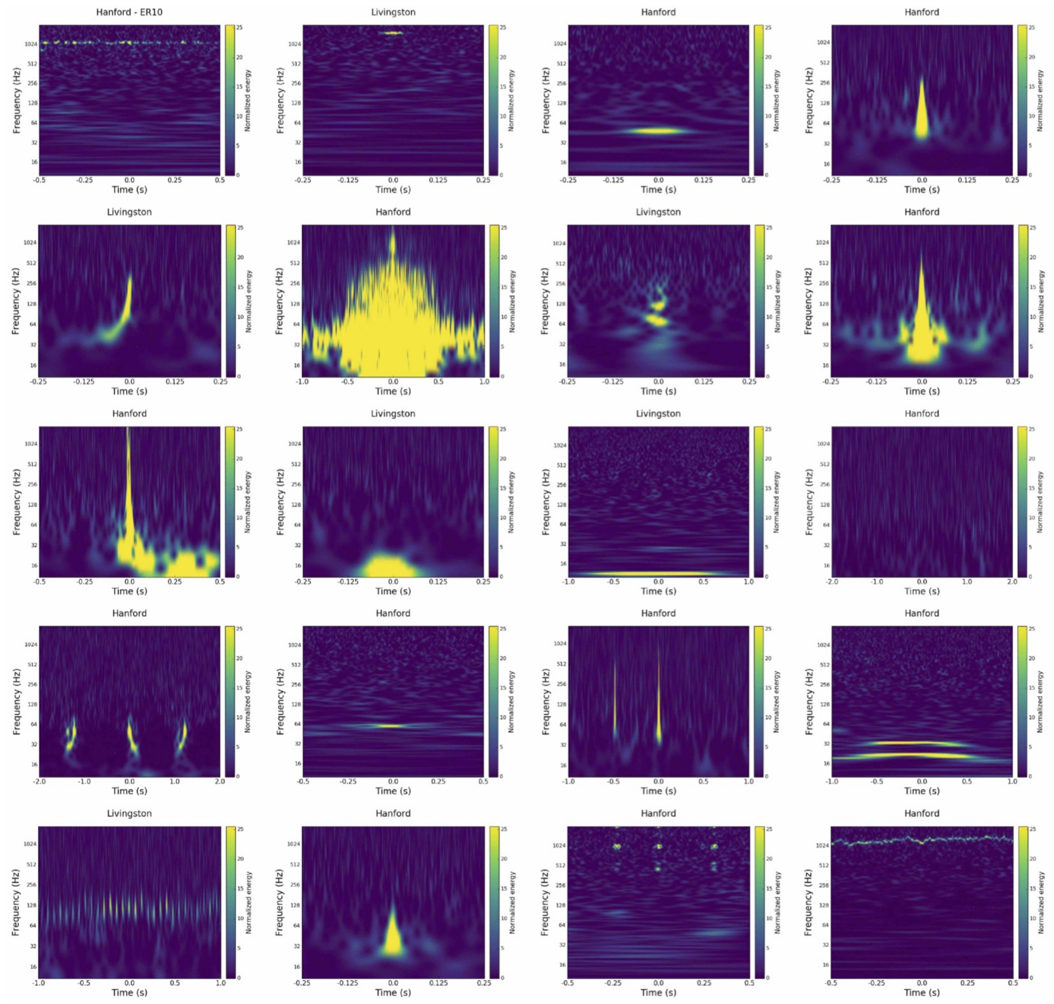
WITH GENERATIVE ADVERSARIAL NETWORKS

Melissa Lopez[1,2], Vincet Boudart[3], Kerwin Buijsman[2], Sarah Caudill[1,2]

1. Institute for Gravitationaland Subatomic Physics (GRASP),Utrecht University, the Netherlands.
2. Nikhef, Amsterdam, the Netherlands.
3. Institut de Physique, Université de Liège, Belgium.



IGWM 2021



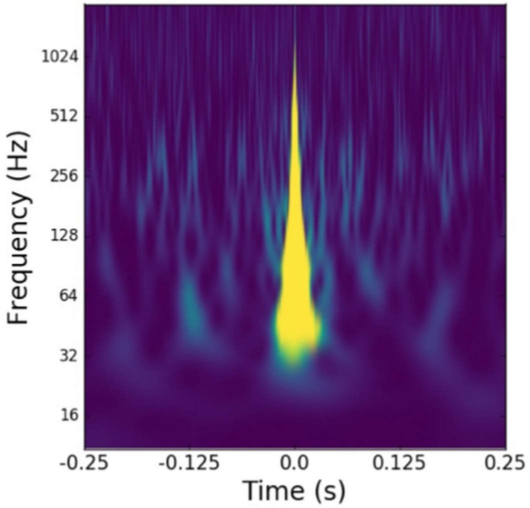
S. Bahaadini. Inf. Sci. 2018

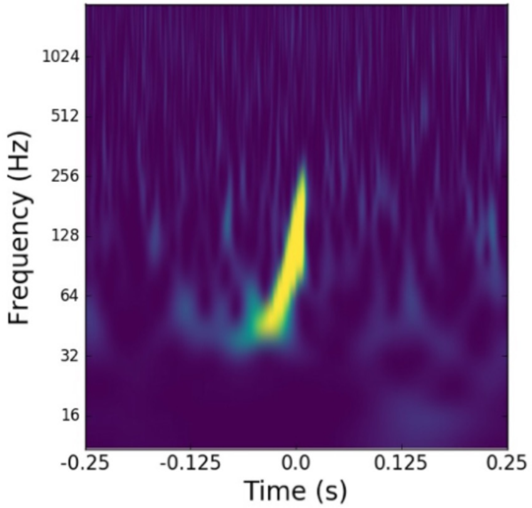
Motivation

- Better inclusion of glitch models in analyses.
- Improve the classification of glitches and new glitches.
- Generate realistic populations of glitches for large-scale studies.

Generate glitches in time domain with GANs

Data set





Example of a blip glitch (left) and a high mass BBH

We focus on blips from L1 and H1, O2

- Simple morphology and abundant
- Similar to other GWs

The noise will hinder our Machine Learning algorithm. Can we separate the glitch from the noise?

Use Bayes Wave for "reconstructing" the glitch

Bayes Wave (BW) dilemma

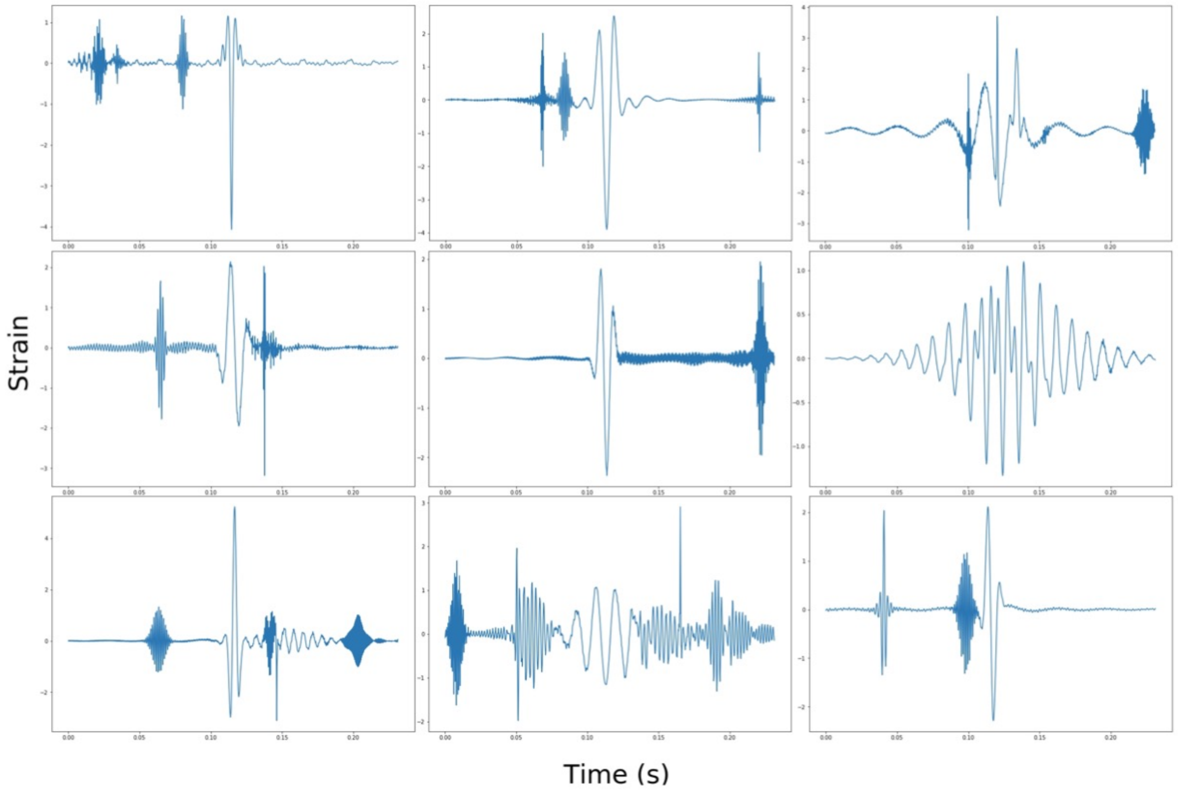
BW is based on wavelet transform

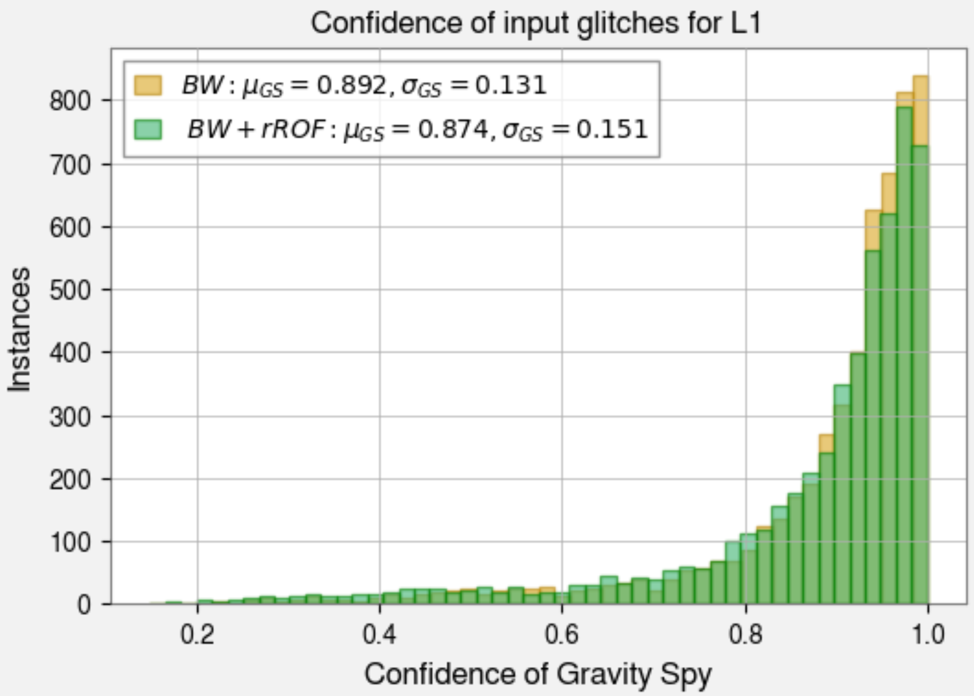
1) We select glitches with high confidence according to Gravity Spy (GS) classifier (ML-based).
2) We reconstruct the glitch with BW.
3) We check its quality with GS.

- In the process we lose more than 50% of the data.
- Even more data is lost for other types.
- Still high frequency noise → use rROF for denoising

Torres-Forné, Phys. Rev. D, 2018

Examples of bad reconstruction with BW





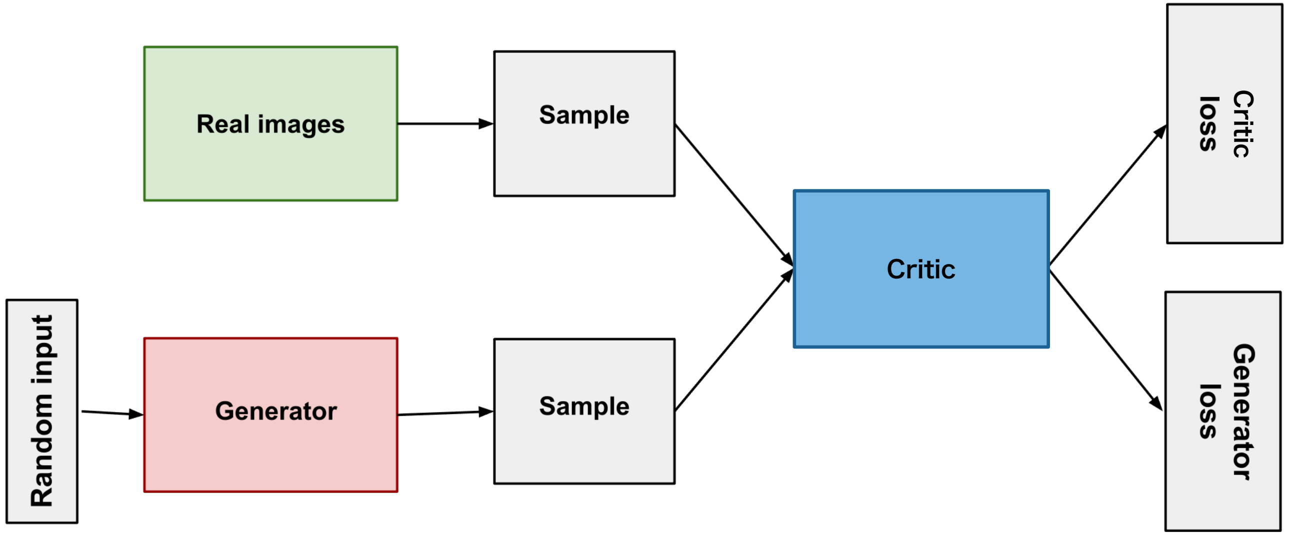
Input blip glitches: L1 example

We "denoise" them with rROF method.

"Light" denoising not to lose too much information.

GAN input: blips > 90% GS confidence.

Generative Adversarial Networks



- Used to learn the underlying distribution of the data
- Inspired by Game Theory: game with 2 networks
- Use Wasserstein loss: critic till optimality
- Very unstable process
- Penalize the network to stabilize it

Network employed: CT-GAN (Wei, ICLR 2018)

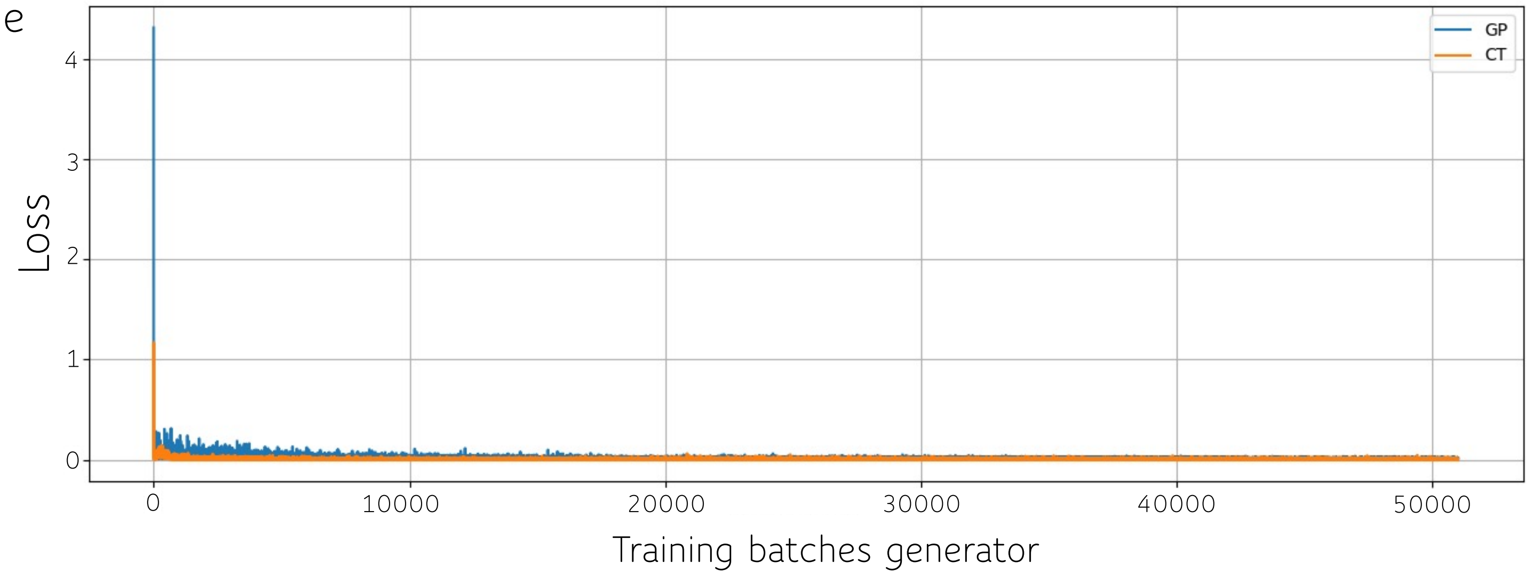
CT-GAN: GP + CT with Dropout

Some intuition from the experiments:

- Gradient Penalty (GP): balances the loss of the critic and generator
- Consistency term (CT): regularizes the generator.
- Dropout: regularizes the critic.

Both terms tend to zero when the network is stable.

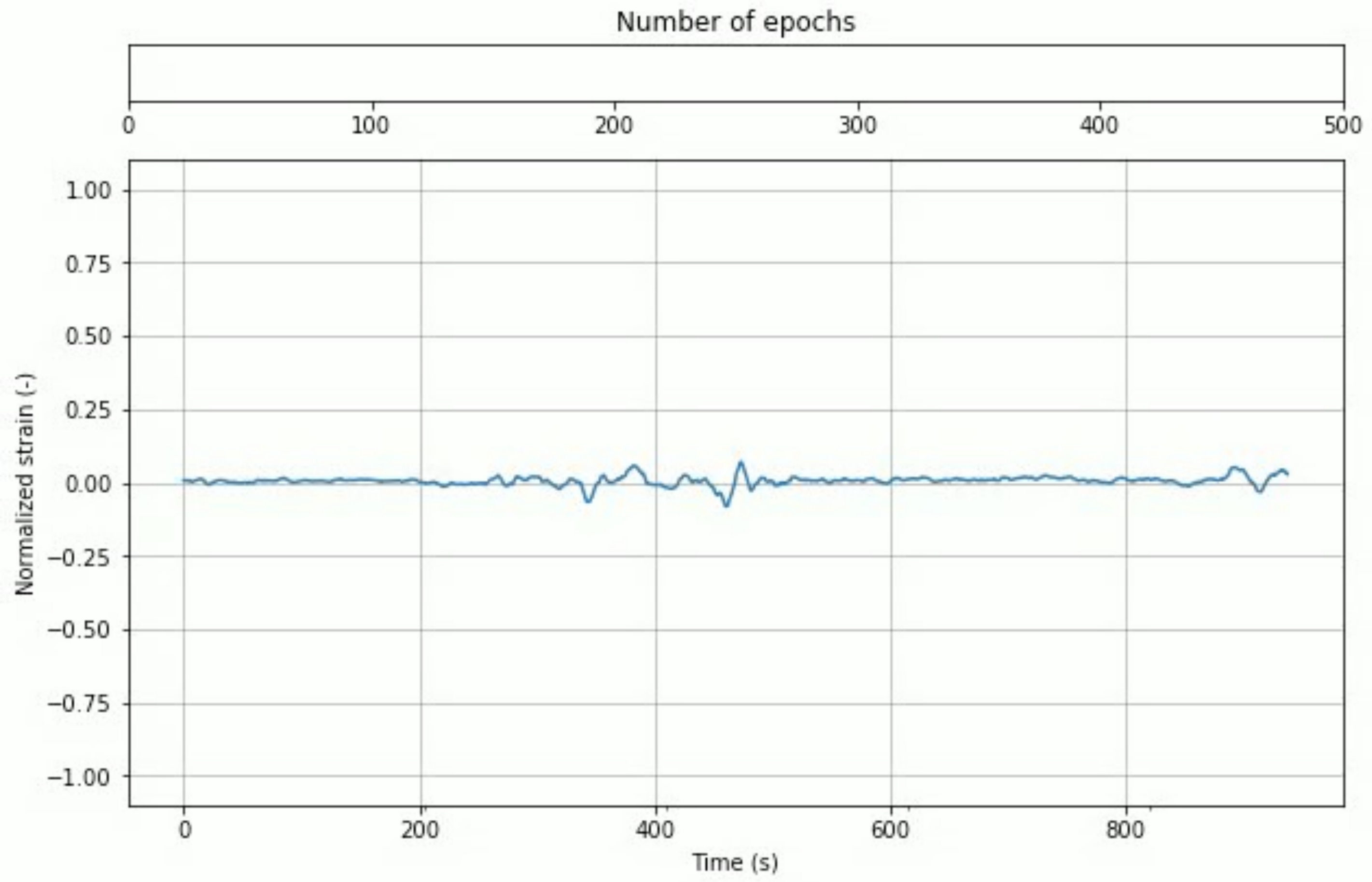
GP and CT loss for each training batch for L1 blips



Note: the generator is trained 5000 times, while the critic 500

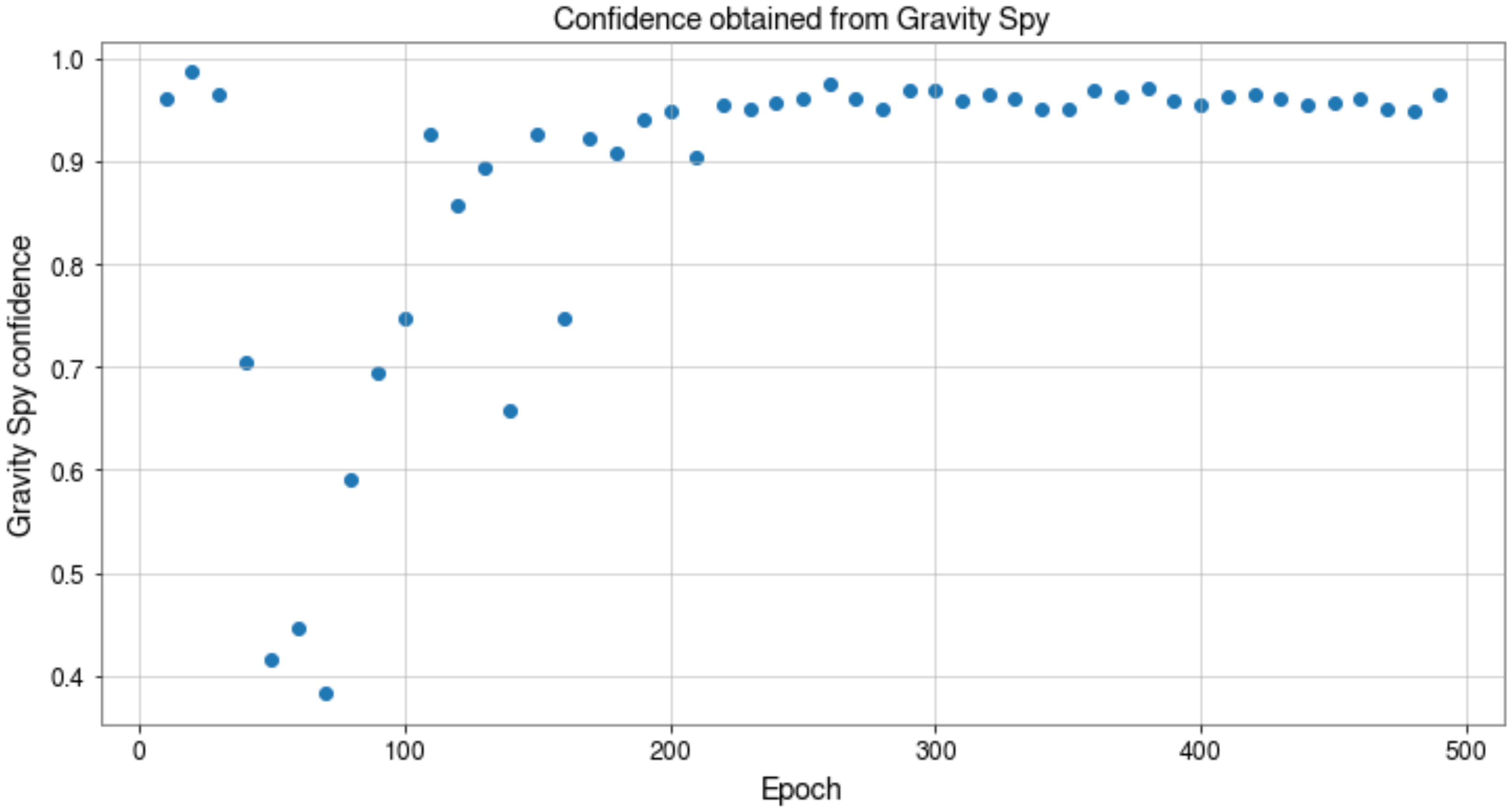
CT-GAN evolution through epochs for H1

- Generator improves quickly
- Non-smooth peaks come from input data



CT-GAN evolution measured by Gravity Spy in L1

Every 10 epochs we generate a fake glitch and measure it with Gravity Spy, to have an extra measure of the performance of the network.



Building a fake population of blips

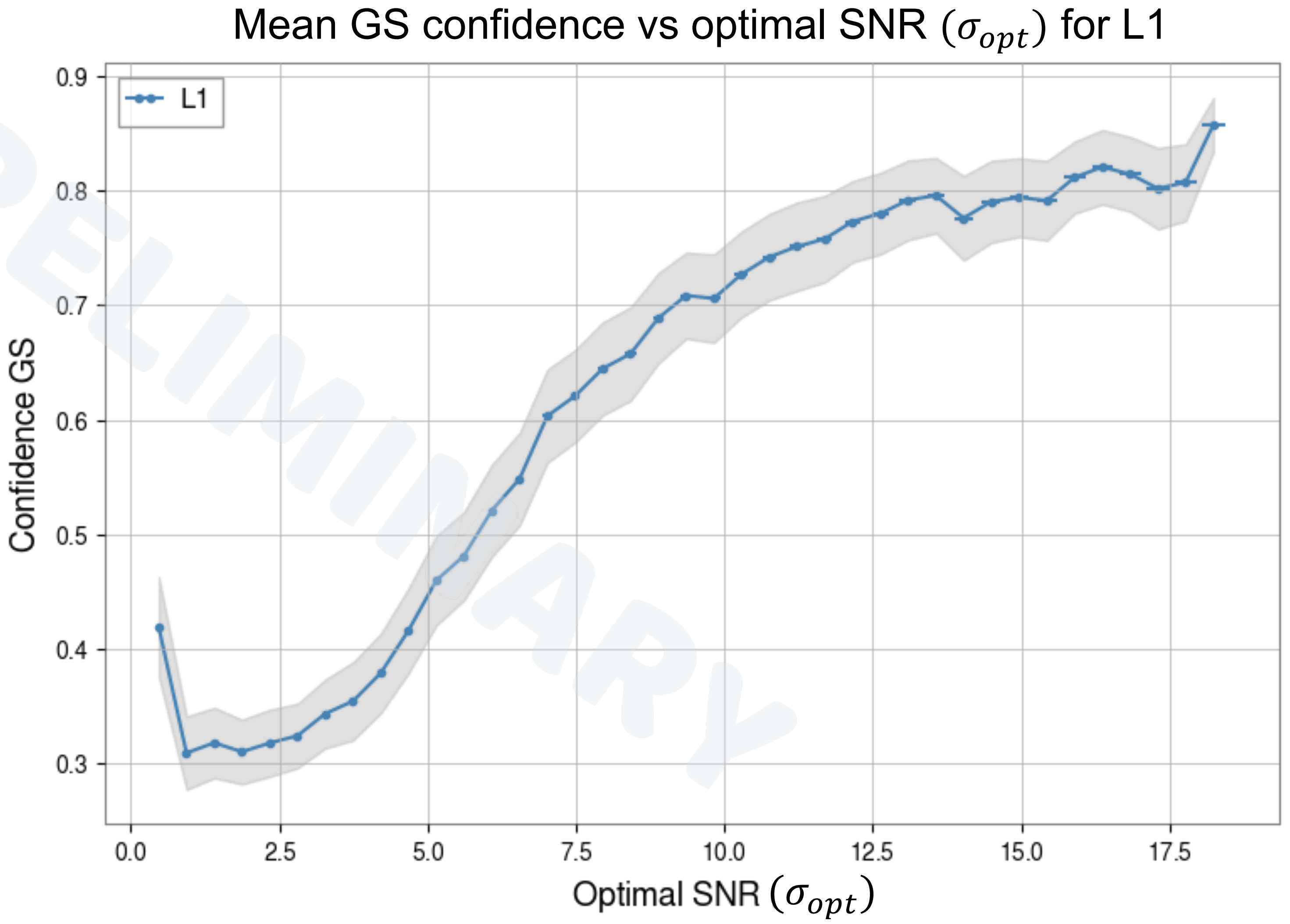
Use α to control σ_{opt}

$$\sigma_{opt} = 4\int_{f_{min}}^{f_{max}} \frac{|\tilde{h}(f)|^2}{S_n(f)}\,df \quad \longrightarrow \quad \sigma_{opt} = 4\alpha\int_{f_{min}}^{f_{max}} \frac{|\tilde{h}(f)|^2}{S_n(f)}\,df$$

Generation of 358 samples, < 1s.

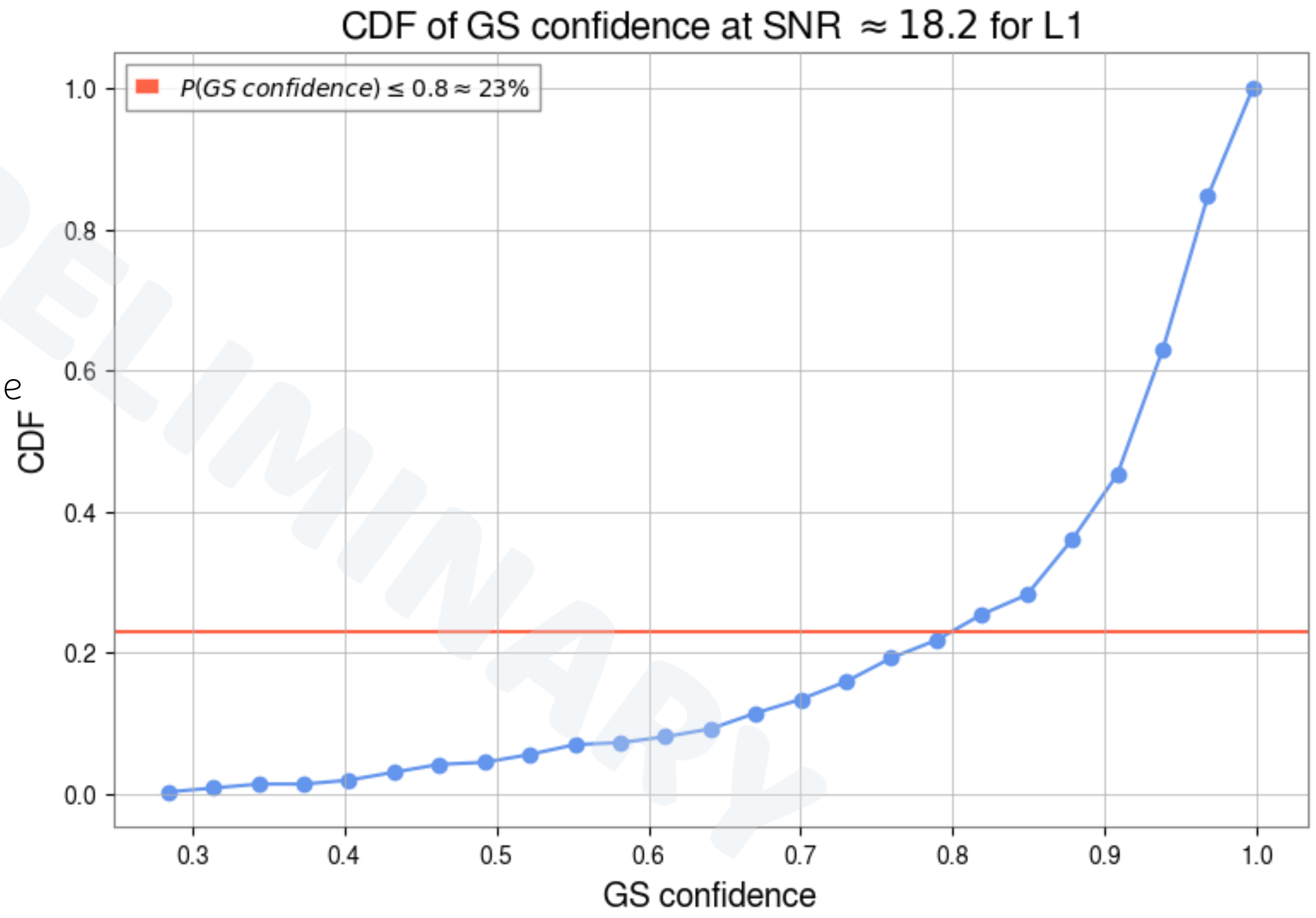
The glitches are injected in background noise and are normalized according to the scale factor α.

Previously selected with GS label 'Blip'.

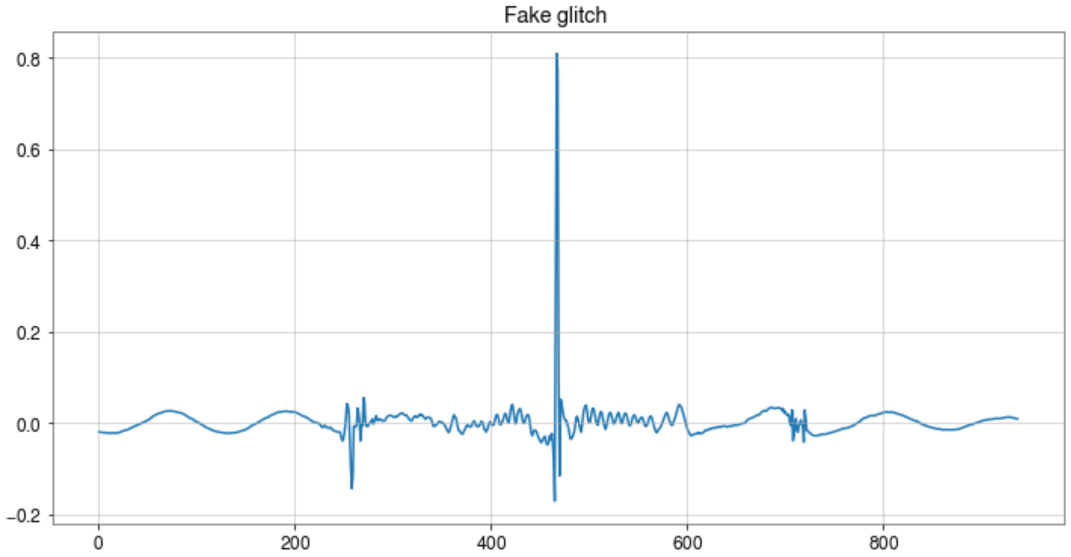


Building a fake population of blips

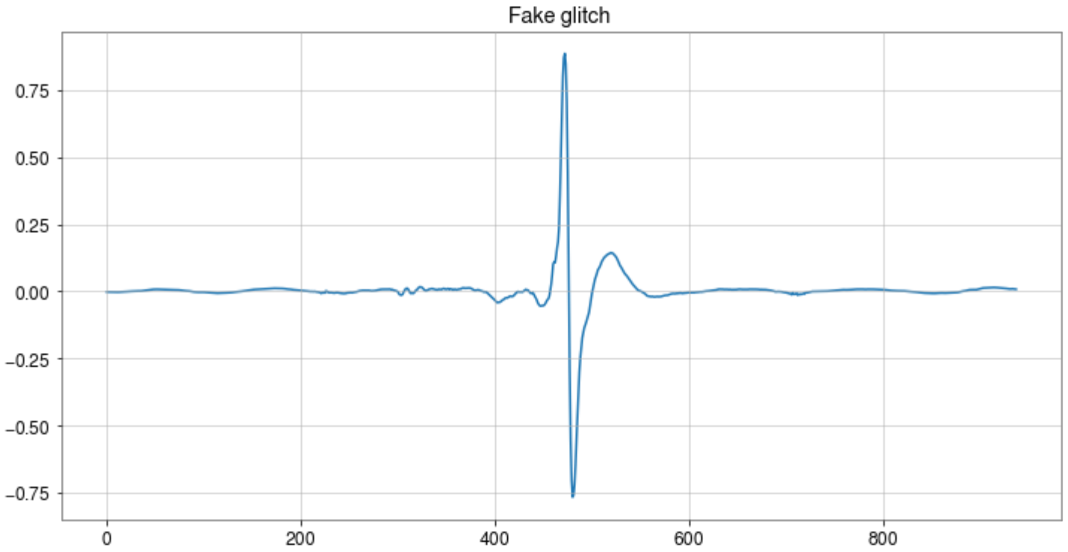
- High GS confidence: good performance
- Low GS confidence: imperfect data



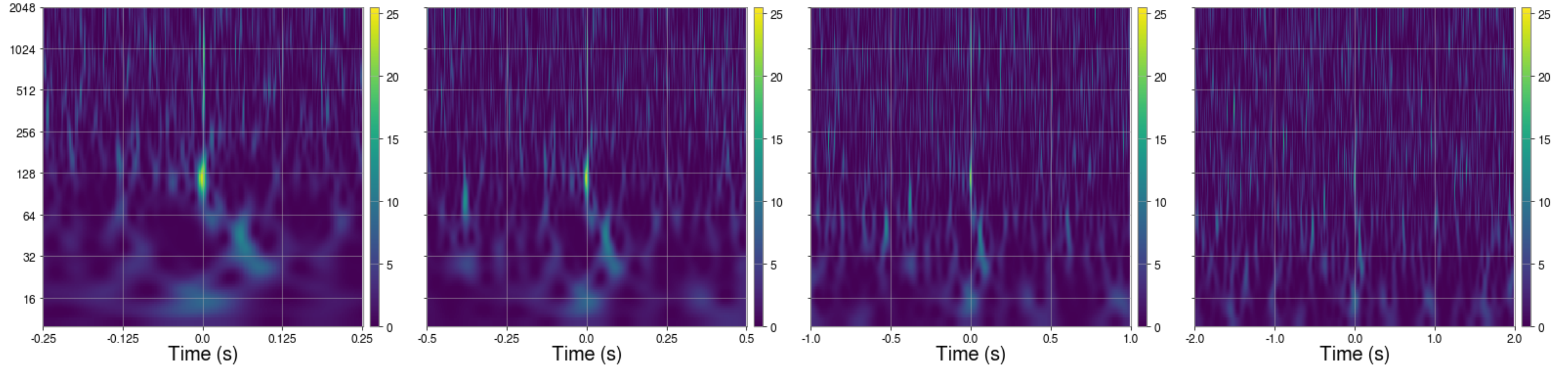
Bad fake glitch



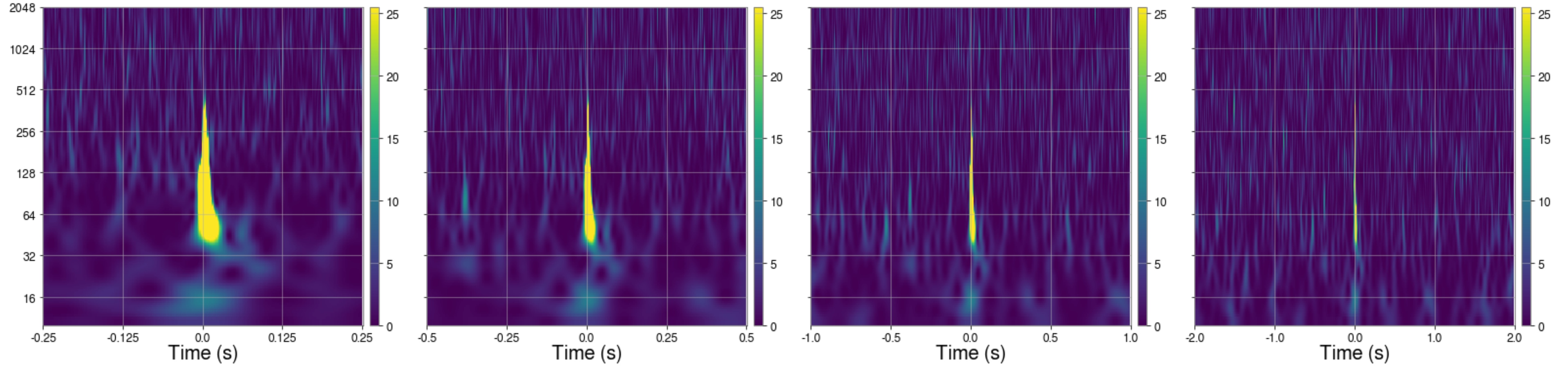
Good fake glitch



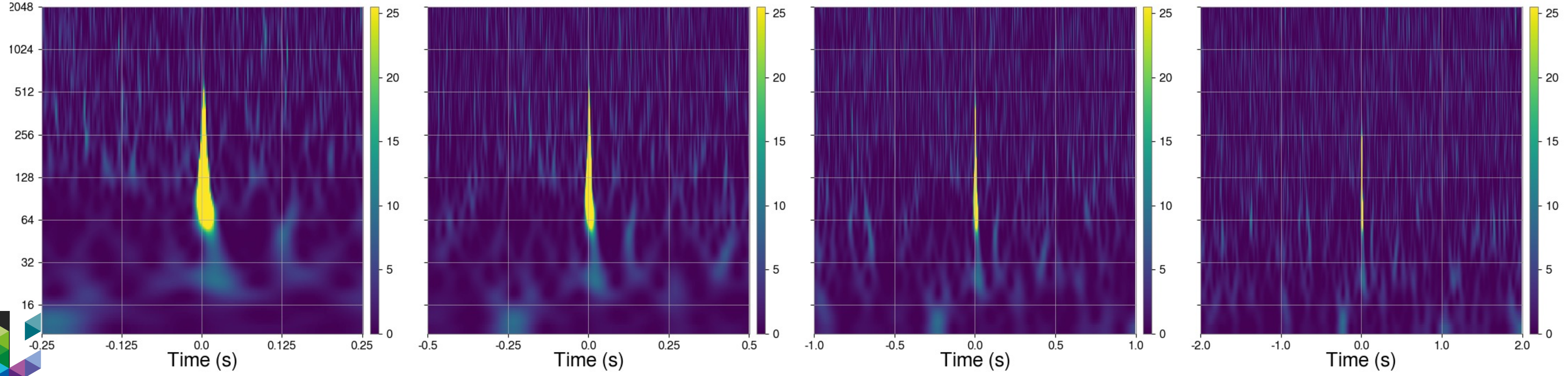
Bad fake glitch

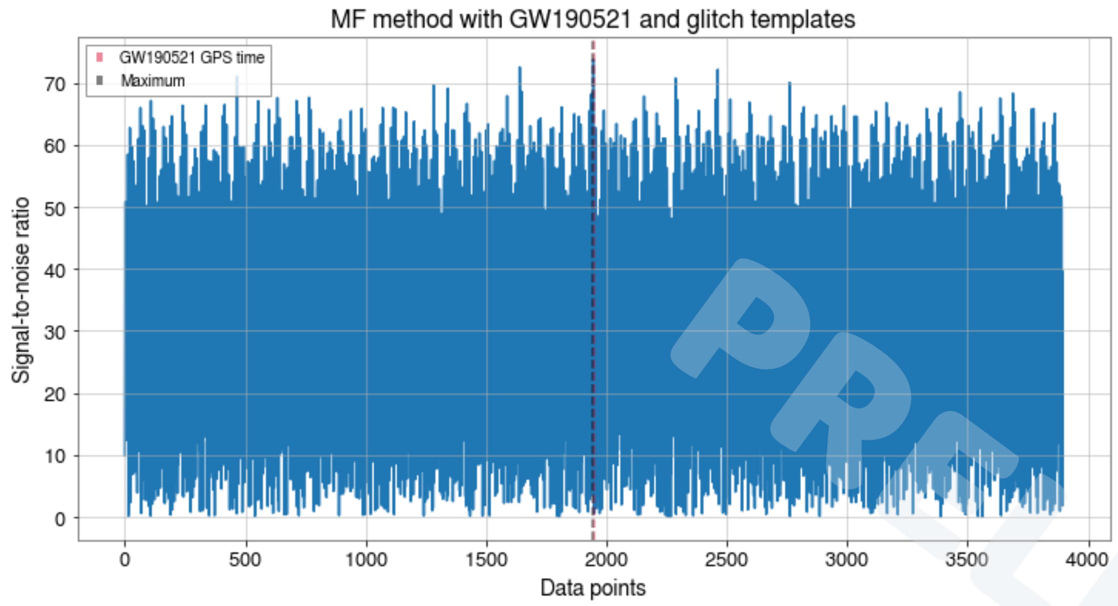


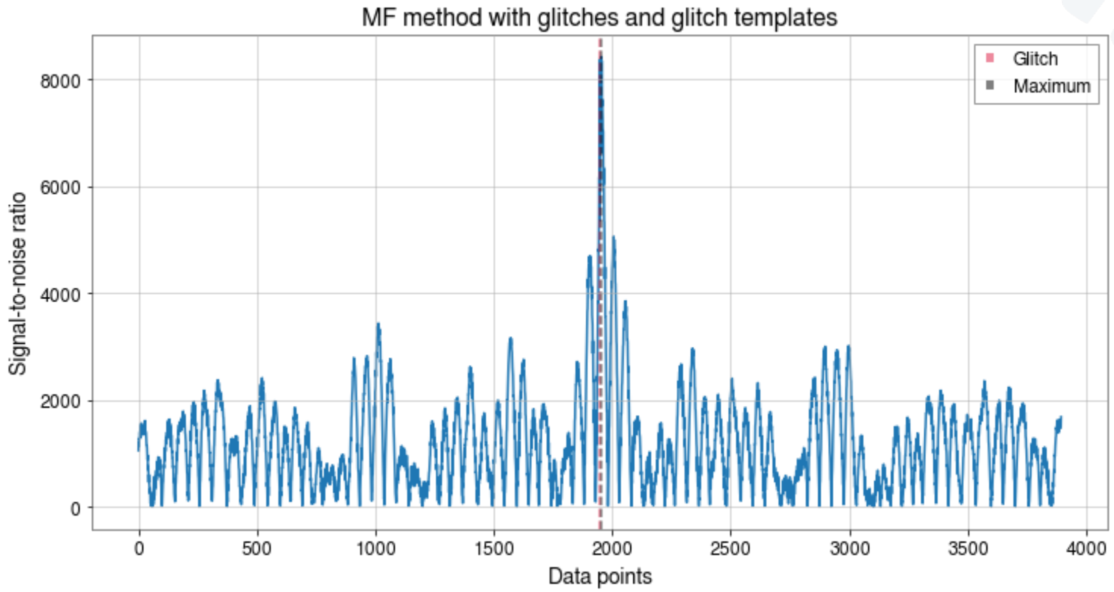
Good fake glitch



Real glitch

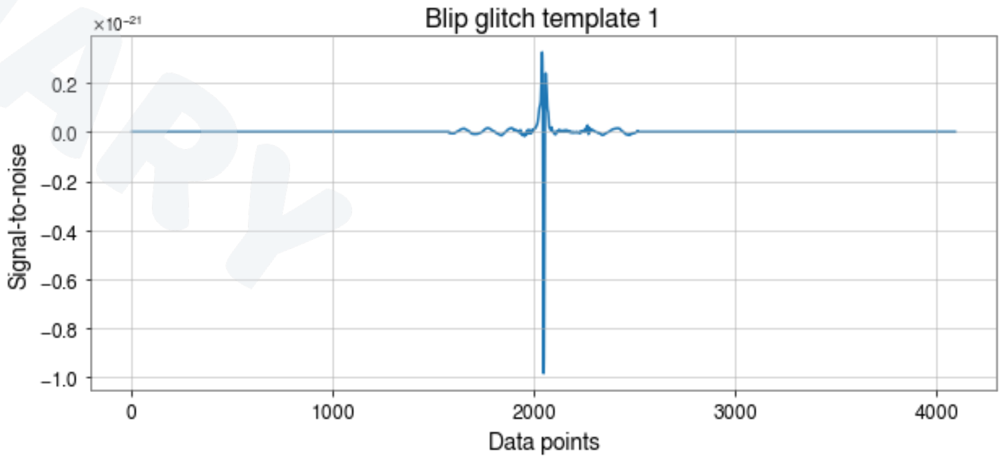






Towards a glitch bank: a small MF test

Idea: glitches match glitch templates better
GW match GW templates better



Conclusion and future work

- We can generate blip glitches.
- GANs need a lot of data, move to O3.
- Constructing a blip bank (in progress)

- BW reconstruction needs to be improved.
- Generalize to other types of glitches.
- Construct a full pipeline for glitch bank.

Thank you! Questions?