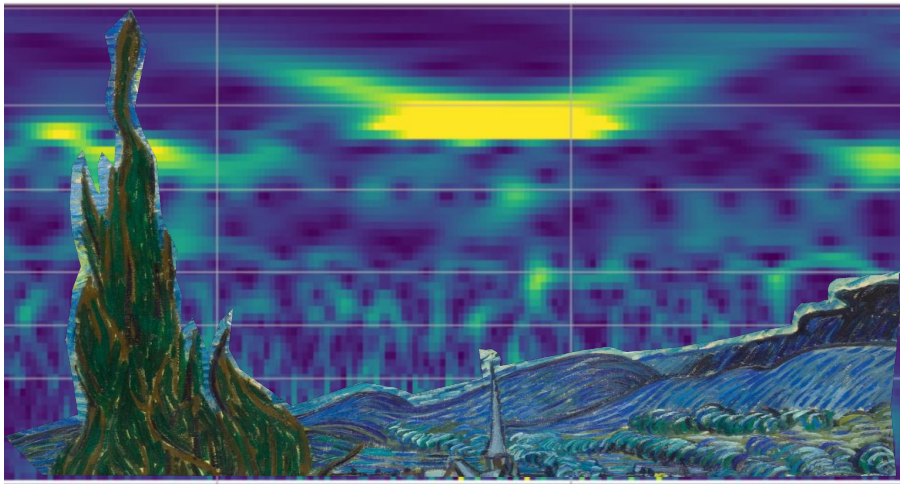
GlitchFlow, a Digital Twin for transient noise in Gravitational Wave Interferometers



🎤***Francesco Sarandrea***[1,2], *Lorenzo Asprea*[1,2], *Alessio Romano*[2], *Federica Legger*[1,2], *Sara Vallero*[1,2]

[1]*On Behalf of the Virgo Collaboration,* [2]*INFN Turin*

IGWM2025- Madrid 23/06/2025

Gravitational Waves and Interferometers

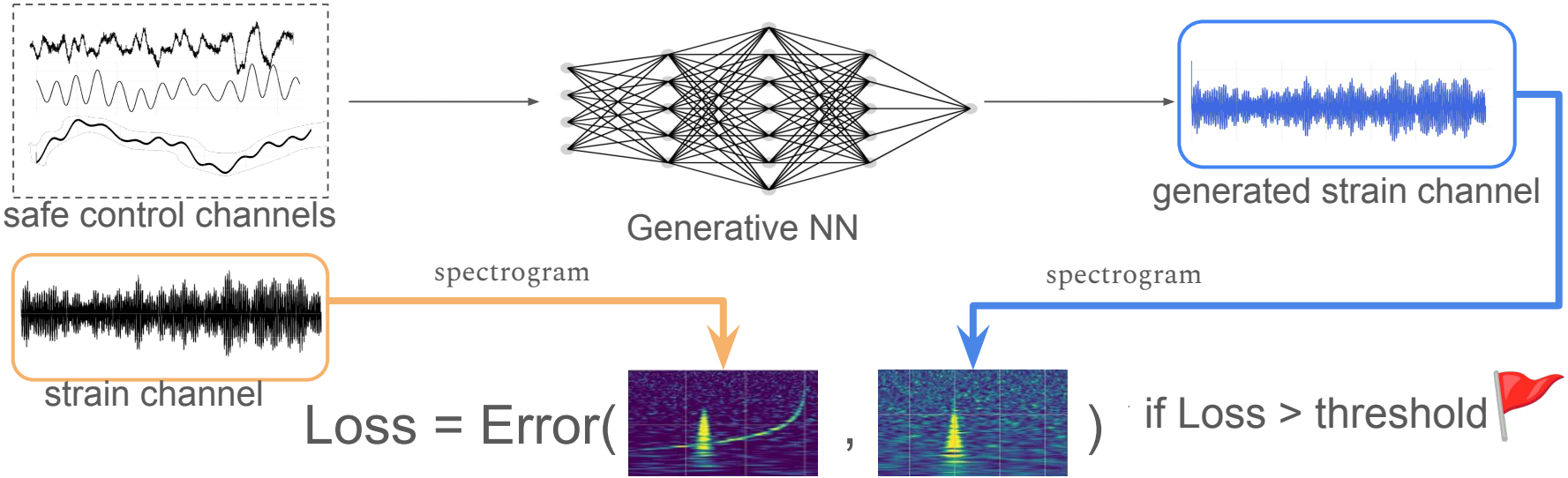
- **Gravitational interferometers** are important because they allow scientists to detect and study **Gravitational Waves (GWs)**.
- **GWs** are ripples in spacetime caused by the acceleration of massive objects, such as **black holes** or **neutron stars**.
- GWs were predicted by **Albert Einstein**'s theory of **General Relativity** but were not directly observed until 2015 with the first detection by the **LIGO – Virgo** Collaboration. (https://doi.org/10.1103/PhysRevLett.116.061102)

GWs are important for:

- **Cosmology and the Early Universe**
- **Fundamental Physics**
- **Multi-Messenger Astronomy**

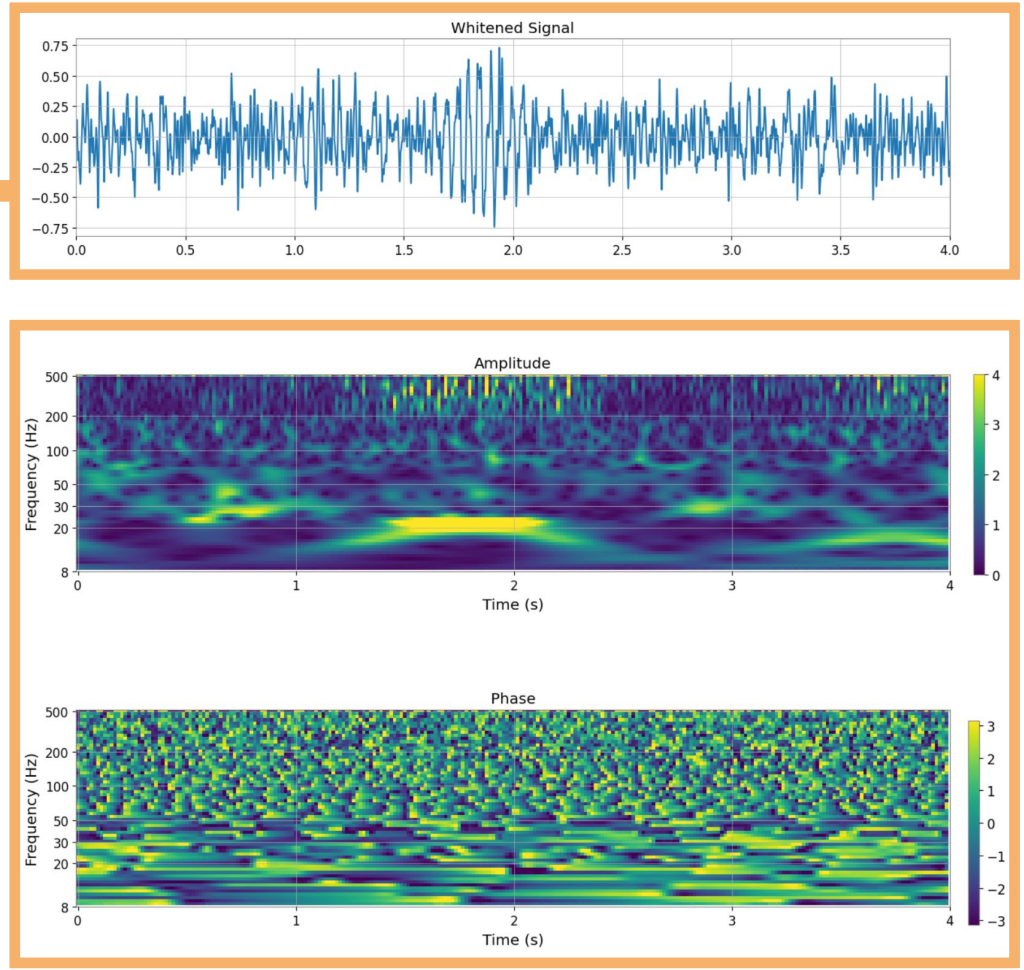
Outline and Goal

- Non-Gaussian transient noise artifacts (aka **glitches**) are one of the most **challenging limitations** in the study of gravitational-wave interferometer
- We **map glitches from control channels** (uncorrelated with the physical signals) **to the main channel**, in order to **subtract the generated noise from the physically interesting data**
- We use **deep generative models** for the mapping in order to capture **non-linear** structure in the **data**



Each whitened 4s timeseries is converted into two 2D spectrograms, one representing the amplitude and one the phase.

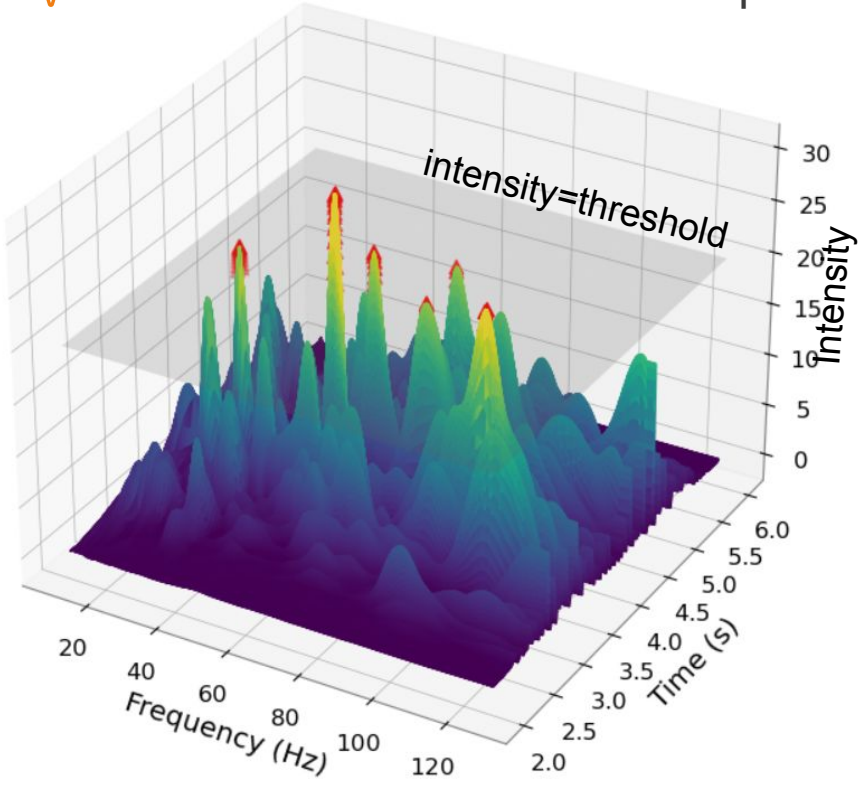
The values of both the amplitudes and the phases are subsequently normalised in the [0,1] interval.



Correlations in Q-Plots: Peak frequencies

Annalisa: channel selection package

CODE RELEASE
https://github.com/interTwin-eu/DT-Virgo-notebooks/tree/main/WP_4_4

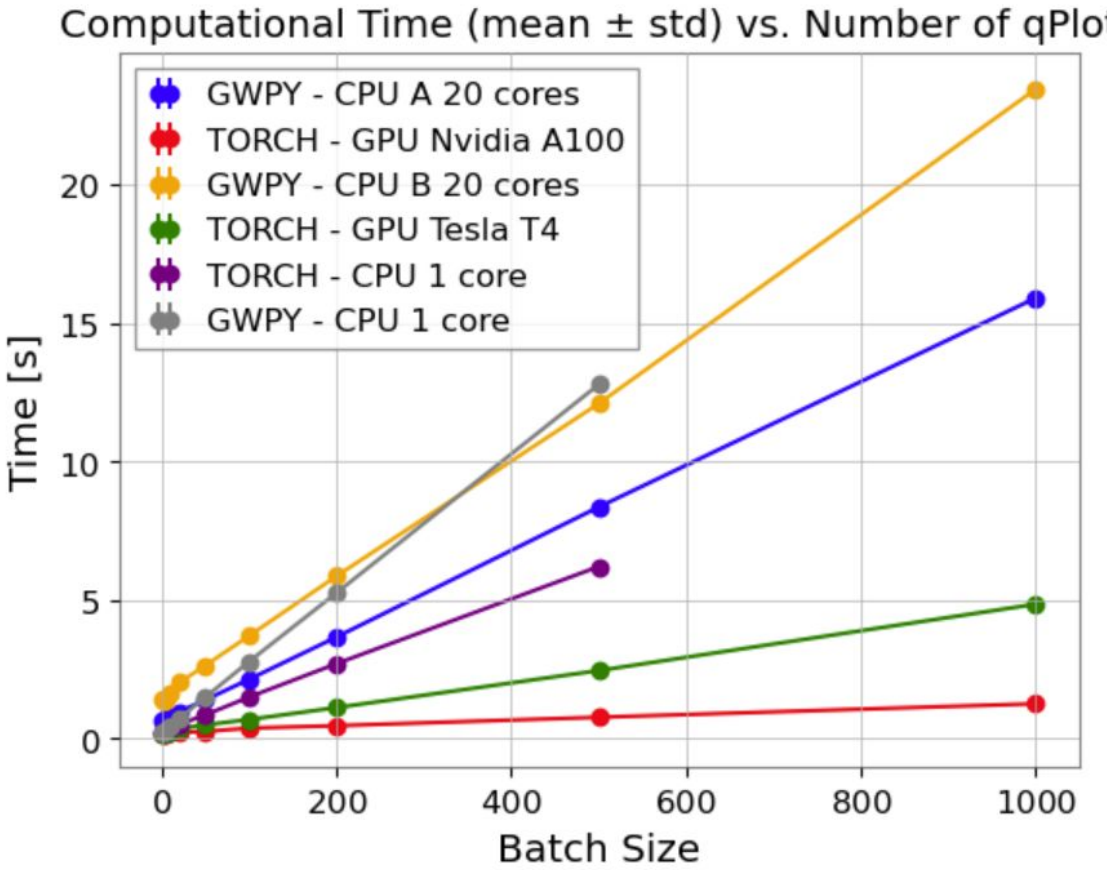


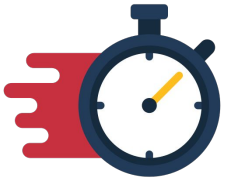
- ***Common peaks***: peaks above intensity threshold at same time
- Apply analysis to each aux channel paired with strain channel
- Calculate correlation coefficient:

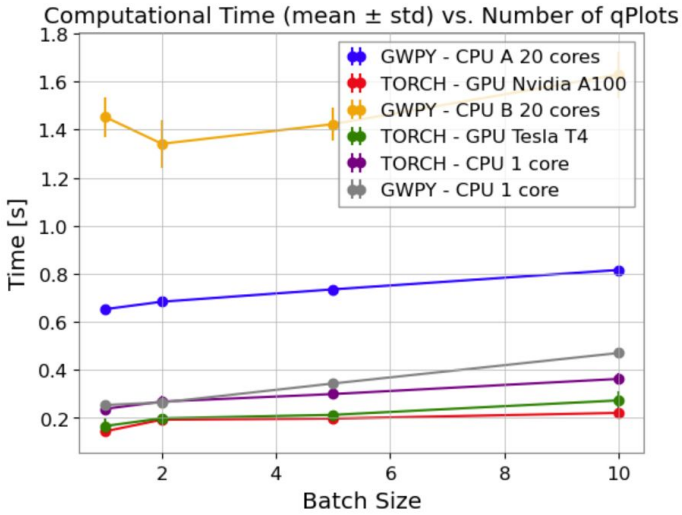
$$\text{Corr\_coeff} = \frac{\textbf{common peaks}}{\textbf{strain peaks}}$$

- Average over 1000 different events classified as Scattered Light with confidence $\geqq$86%.

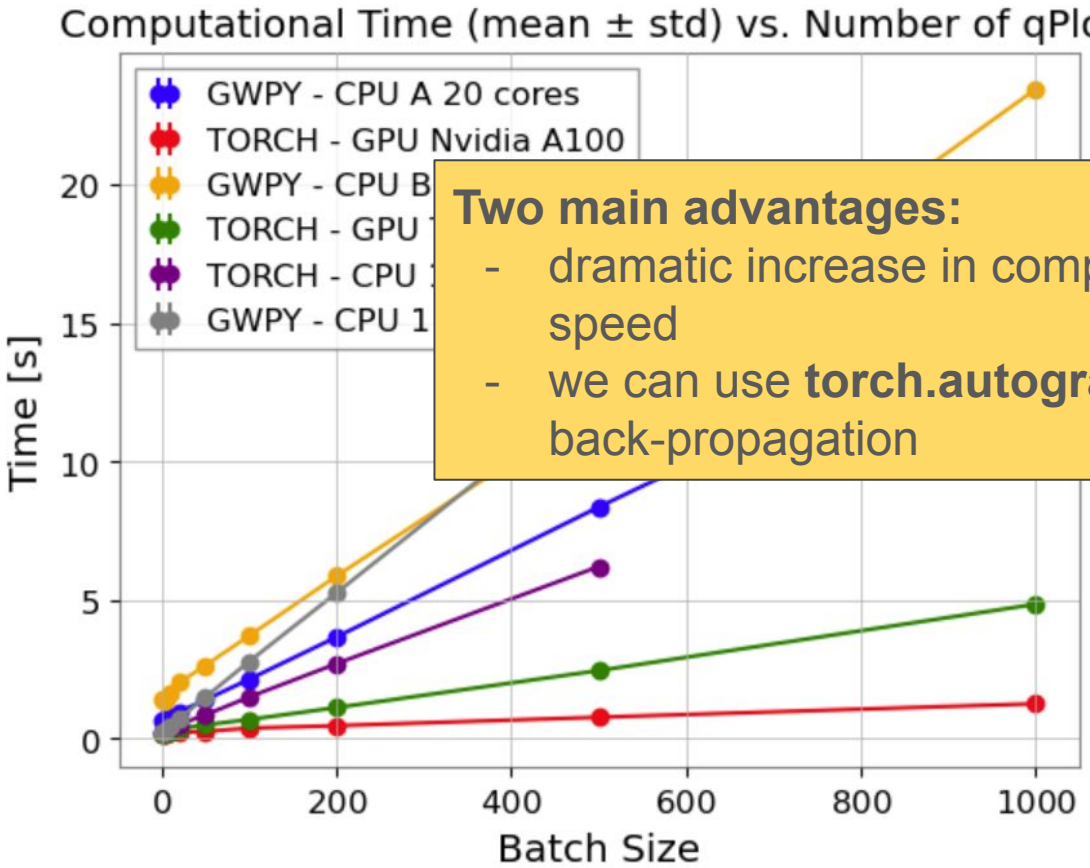
Torch Q-Transform: Speed test





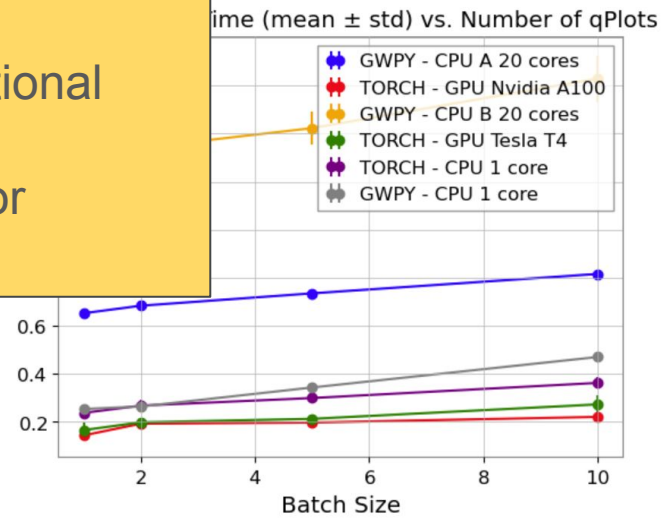


Torch Q-Transform: Speed test



Two main advantages:

- dramatic increase in computational speed
- we can use **torch.autograd** for back-propagation



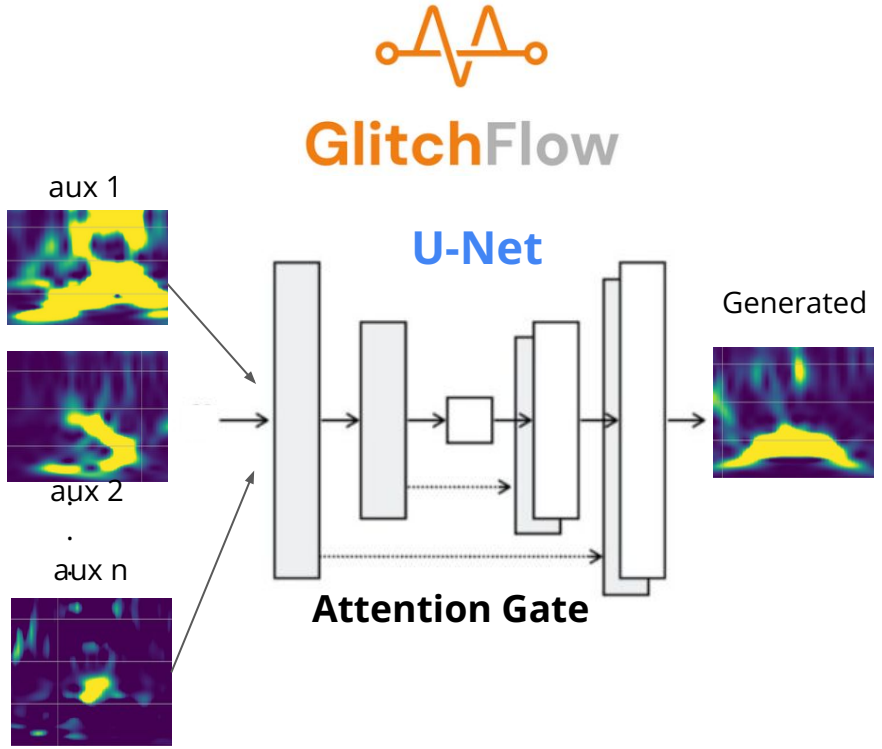
Neural Network & Dataset

Data & Preprocessing:

1. 12K Scattered Light glitches in O3a VIRGO
2. 6s long time series
3. Resampling to 4096Hz & whitening
4. Spectrogram generation in 8-500Hz range
5. Image cropping to 1s frames
6. Normalization

Neural Network: U-Net with residual blocks and attention gates enhanced skipped connections.

Training Loss Function: Structural Similarity Index Measure + L1.



GlitchFlow: https://github.com/interTwin-eu/DT-Virgo-notebooks/tree/main/Release_4.6

Digital Twin Engine integration: Usage Example

Pipeline configuration

User can set via config file:

- **Steps and order** (scan, preprocess, training, inference, visualization)
- **Training parameters** (loss, lr ,num epochs, ...)
- **NN architecture** and **weights** form **model registry**
- Path for **saving and loading** weights, data, results, …
- **Preprocess parameters**, steps and **dataset**

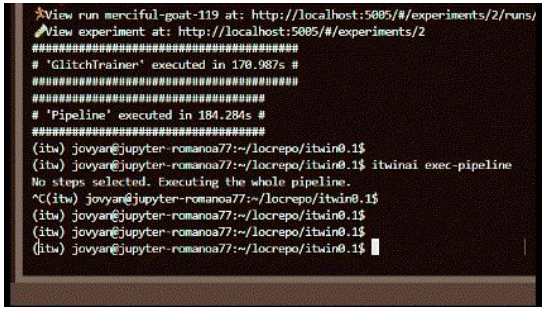
Config file .yaml

```yaml
#config.yaml
training_pipeline: #pipeline name
  _target_: itwinai.pipeline.Pipeline
  steps:
    Splitter:  #step 1
      #every task is implemented as a python class
      _target_: itwinai.plugins.glitchflow.Dataloader.QTDatasetSplitter
      images_dataset: '/path/dataset.pt'
    Processor: #step 2
      _target_: itwinai.plugins.glitchflow.Dataloader.QTProcessor
    Trainer:   #step 3
      _target_: itwinai.plugins.glitchflow.Trainer.GlitchTrainer
      num_epochs: 2
      config:
        _target_: itwinai.torch.config.TrainingConfiguration
        batch_size: 2
        logger:
          _target_: itwinai.loggers.MLFlowLogger #save model
          tracking_uri: 'http://localhost:5000'
```

Pipeline execution

```
$ itwinai exec-pipeline --config-name config.yaml  +pipekey=training_pipeline +pipeline_steps=[Splitter,Processor,Trainer]
```

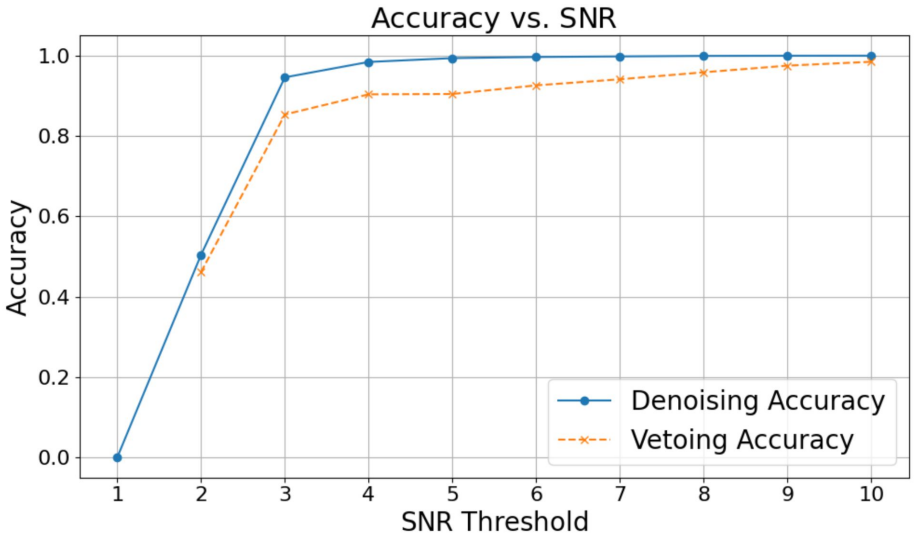
- The pipeline has been implemented as an **itwinai** plugin
- **itwinai** seamlessly integrates the pipeline with logging and model catalog features offered by **mlflow**
- **itwinai** allows for distributed training
- Plugins are modular: reuse and adapt code for efficiency
- Easier for user to configure and adapt pipeline with built in classes

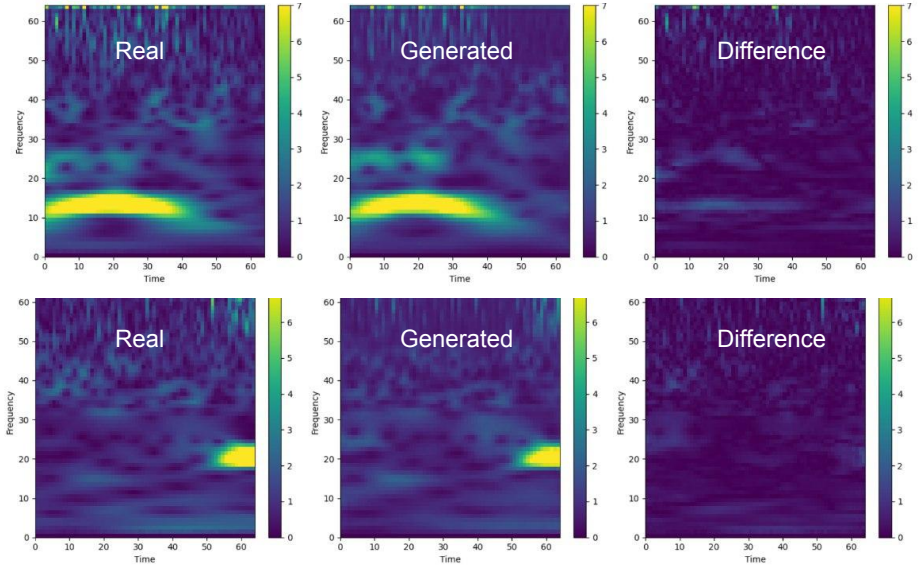


Performance Tests

	SNR 3	SNR 6
Glitch vetoing Accuracy Model is able to correctly generate a glitch above threshold SNR given control channels	**85%**	**95%**

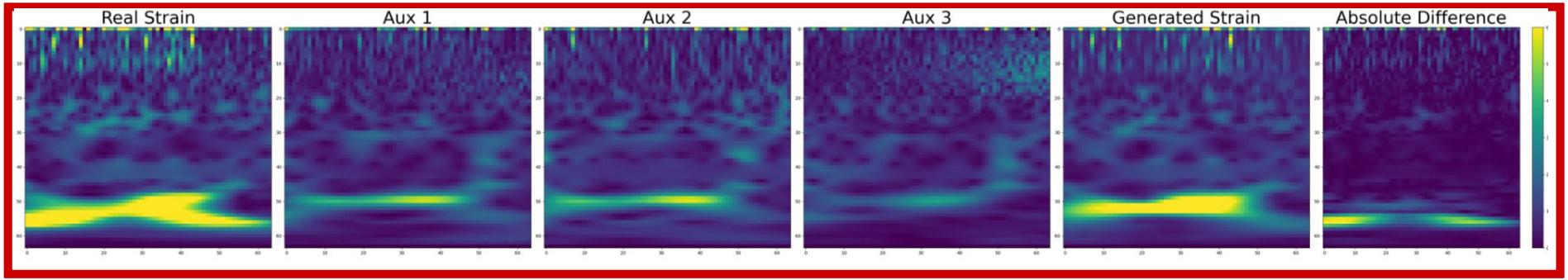
	SNR 3	SNR 6
Glitch denoising Accuracy Model generates glitches with right time and frequency. Noise subtraction has SNR lower than threshold	**95%**	**99%**



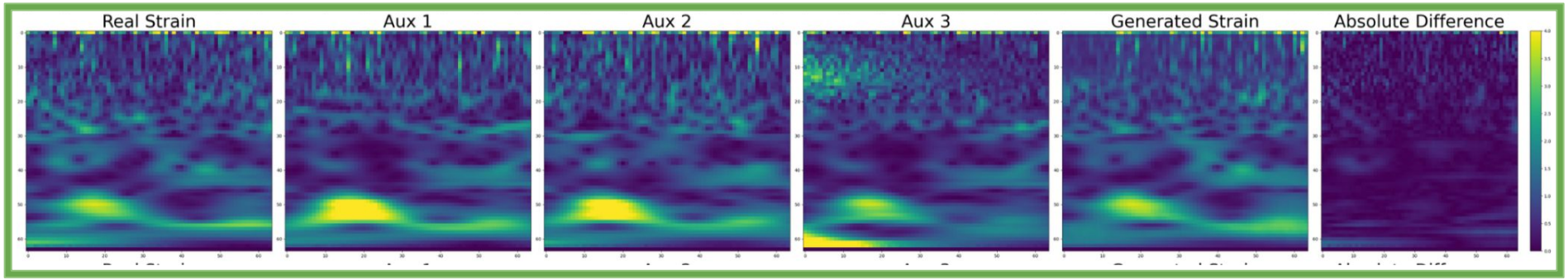


When Does the Model Fail?

Model Fails

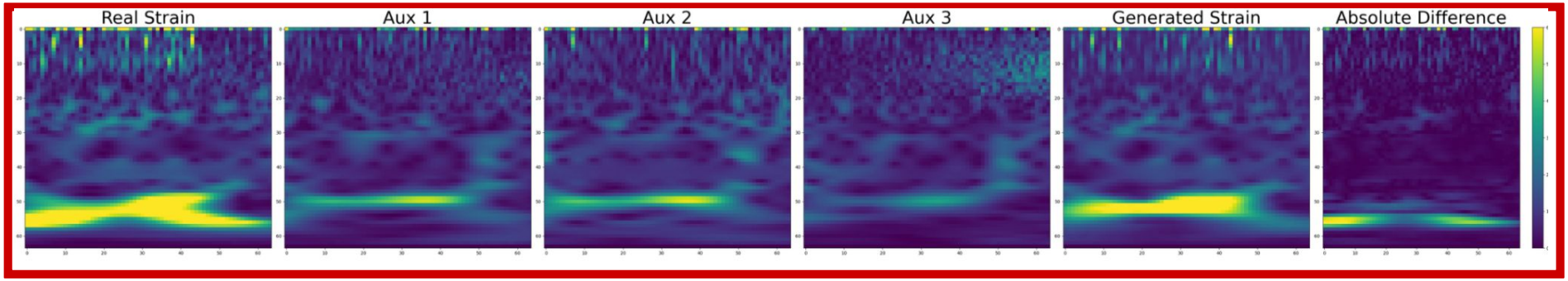


Model Succeeds



When Does the Model Fail?

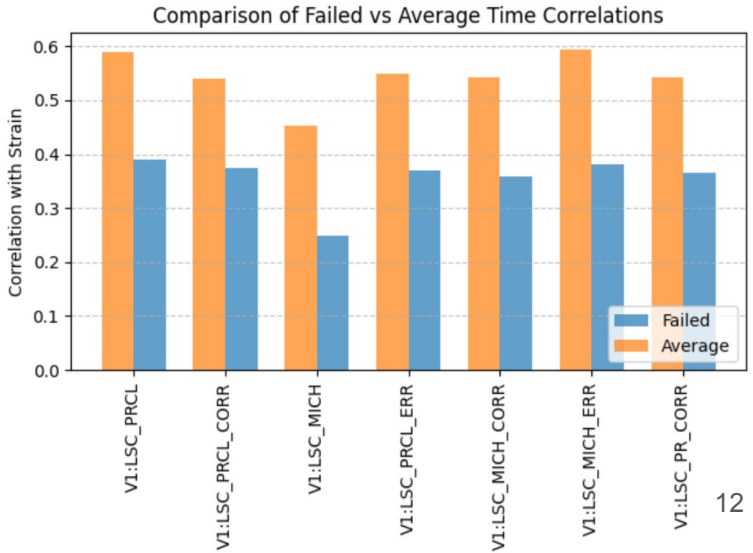
Model Fails



For the events in which de-noising fails, we measure a lower correlation between the auxiliary channels and the strain

Possible solutions:

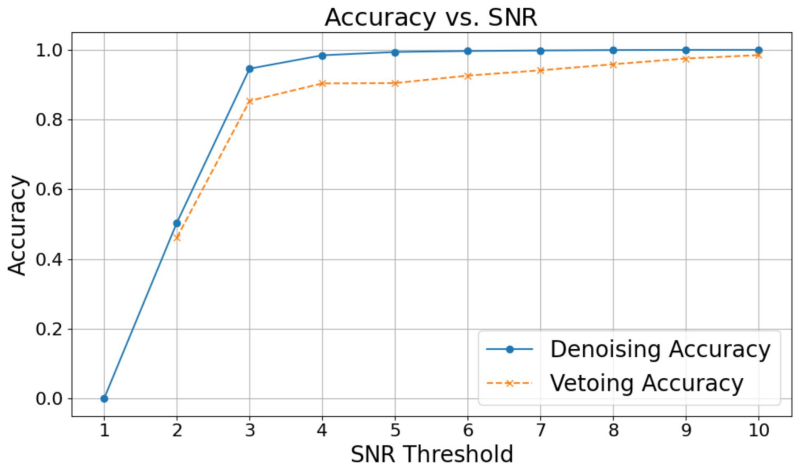
- train on more events,
- use a larger set of auxiliary channels.



Conclusions & Outlook

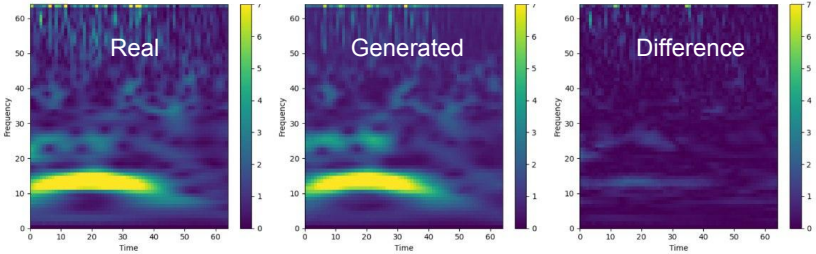
ACHIEVEMENTS

- Developed a **pytorch-based Q-Transform** which allows efficient implementations for NN applications.
- The DT successfully implements a **vetoing/denoising** strategy for scattered light glitches.
- **Vetoing accuracy higher than 90%** for glitches above SNR=6.
- **De-noising accuracy higher than 98%** for glitches above SNR=4.

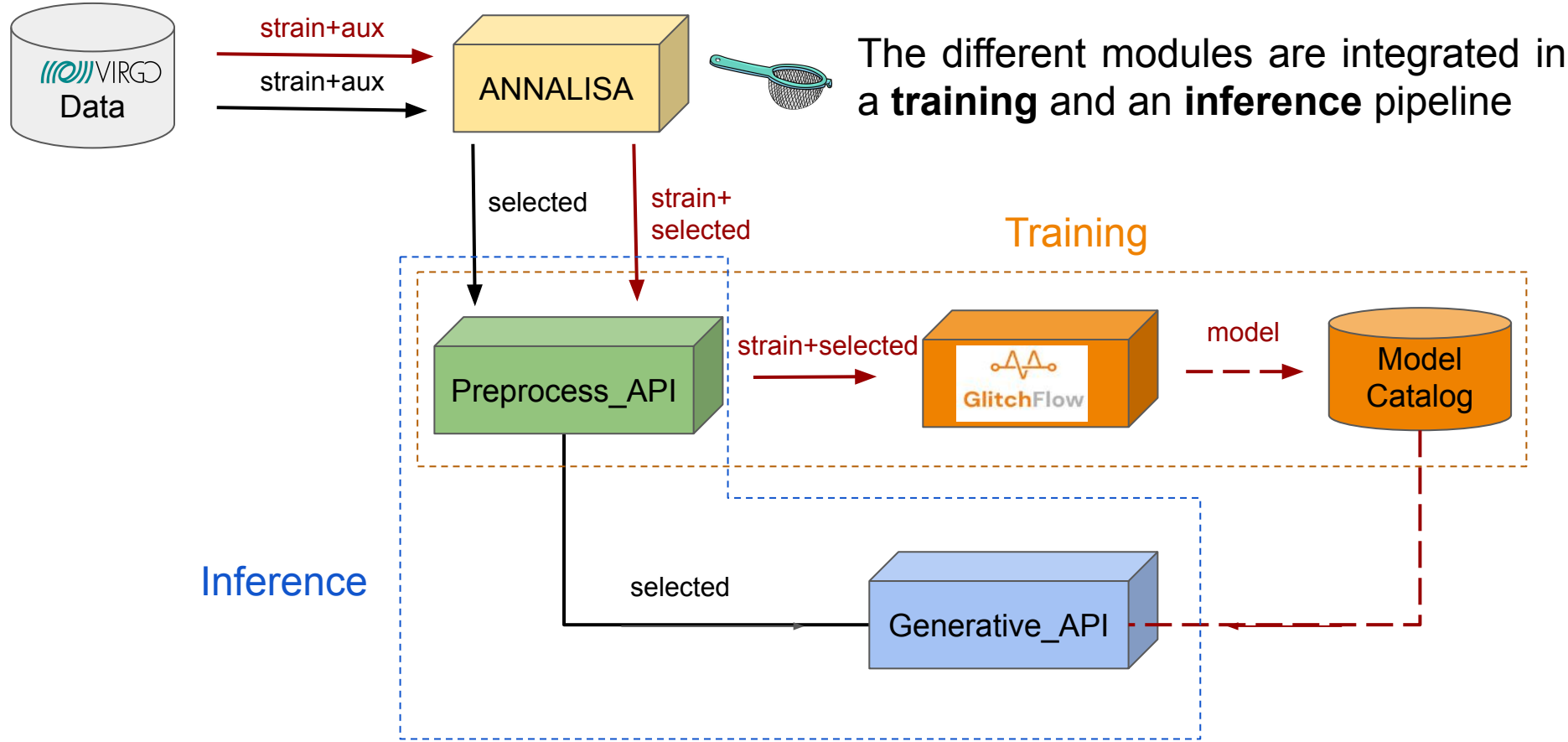


FUTURE WORK

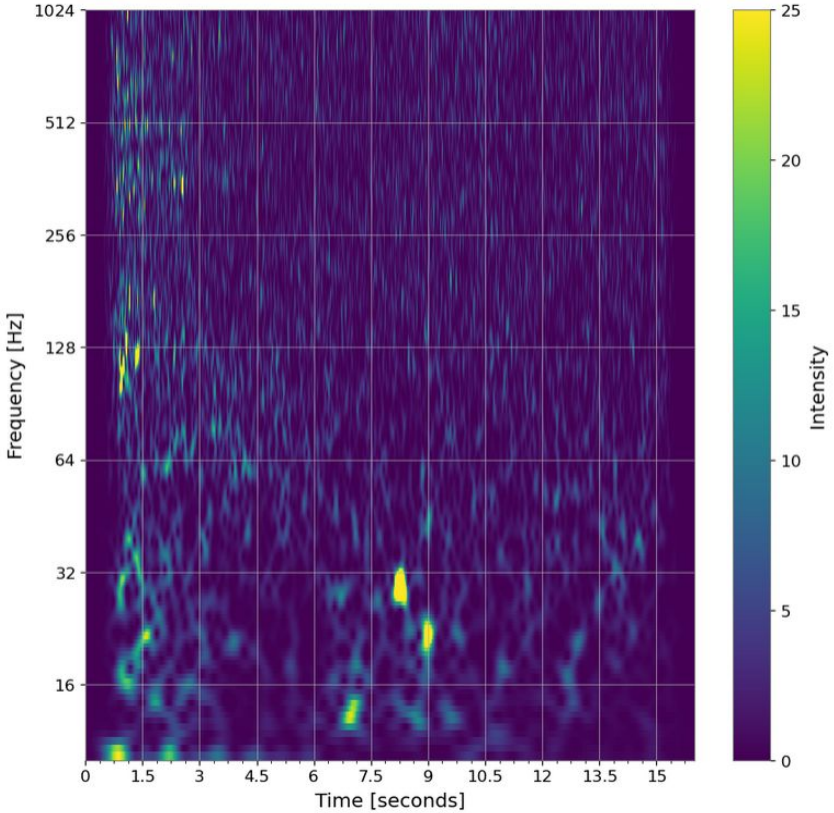
- Study different subsets of auxiliary channels.
- Perform training and inference on **O4 data.**
- Improvement of whitening techniques to avoid data loss.
- Study **de-noising methods in the time domain** to integrate GlitchFlow with the pipelines for parameter estimation.
- Test the de-noising on real and simulated GWs.



Backup Slides


VIRGO
Data
strain+aux
strain+aux
ANNALISA
The different modules are integrated in a **training** and an **inference** pipeline
selected
strain+
selected
Training
Preprocess_API
strain+selected
GlitchFlow
model
Model
Catalog
Inference
selected
Generative_API

Qtransform



$$X(\tau, \phi, Q) = \int_{-\infty}^{+\infty} x(t) w(t - \tau, \phi, Q) e^{-i2\pi\phi t} dt$$

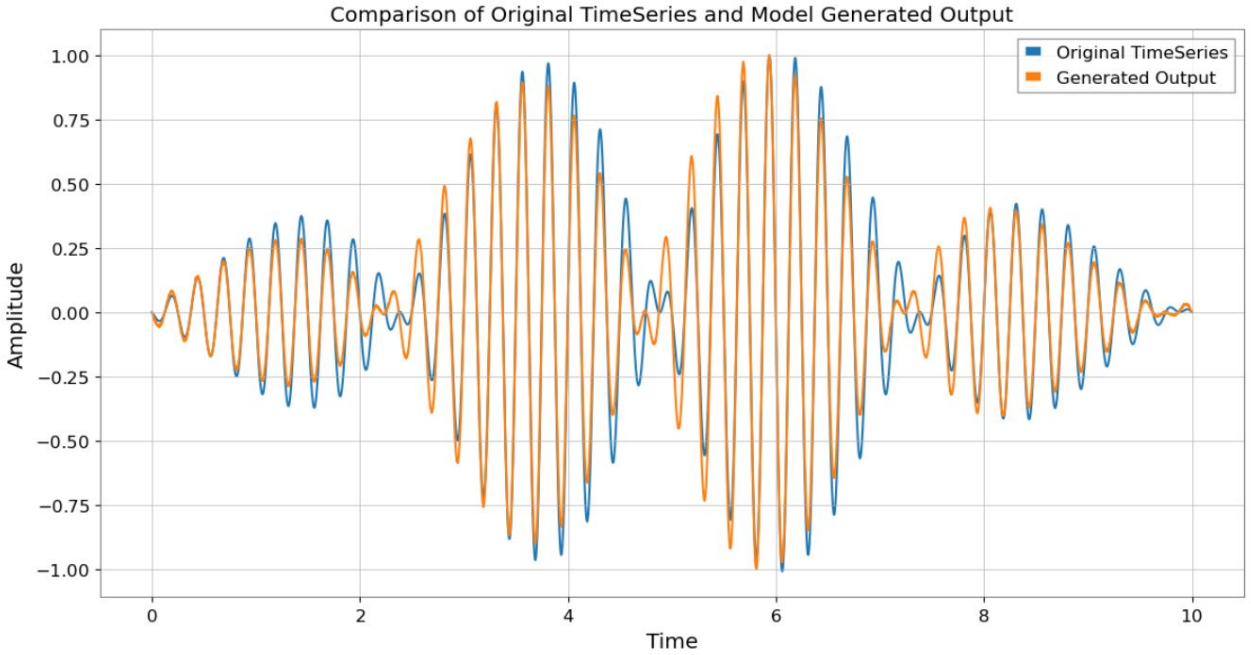
$$w(t - \tau, \phi, Q) = \frac{C}{\sqrt{2\pi}\sigma_t(Q, \phi)} \exp\left[- \frac{(t - \tau)^2}{2\sigma_t(Q, \phi)^2} \right]$$

$$\sigma_t(Q, f)^2 = \frac{Q^2}{8\pi^2 f^2} \qquad Q = \frac{f}{\delta f}$$

Issues:

- Potentially excellent for ML applications, but hard to use as it is **not invertible** and results cannot be used in downstream pipeline (e.g. parameter estimation)
- GWPy's implementation is **too slow** for heavy computations and (some) low-latency pipelines

Q-transform inversion using Transformers



$$\mathrm{SCL} = \frac{|||S_{target}| - |S_{predicted}|||_F}{||S_{target}||_F}$$

Average Pearson Correlation=82%

Future Outlook: Q-transform inversion

All the pipelines for parameters' estimation take one-dimensional timeseries as input. Our tools for the subtraction of glitches output a CQT instead. For this reason we want a Neural Net (NN) which recovers the timeseries from the cleaned QT.