Machine Learning in LIGO

Tabata Ferreira

Postdoctoral Researcher, INPE

Siddharth Soni

Asst Project Scientist, UCR

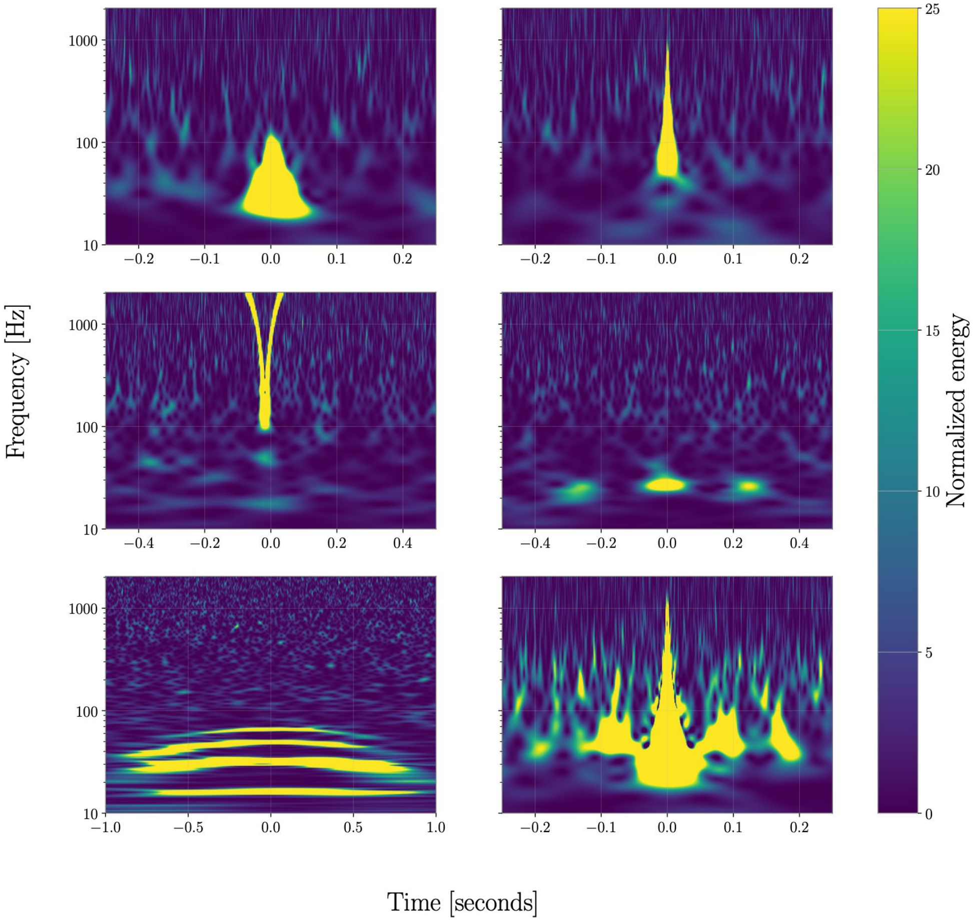
IUCAA Detector Characterization Workshop
Dec 2025
Pune, India

Overview

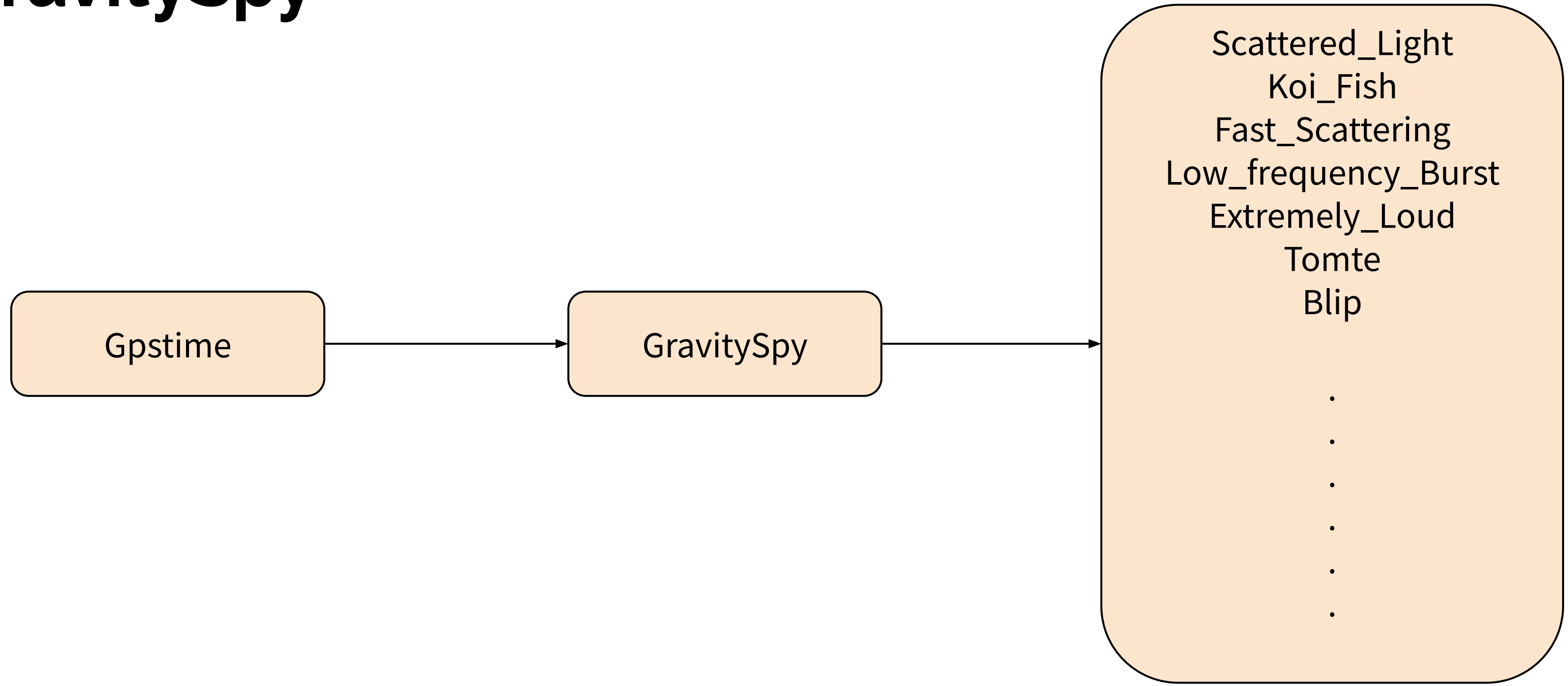
- Introduction
- GravitySpy
- t-SNE
- GW-YOLO
- Conclusion

GravitySpy

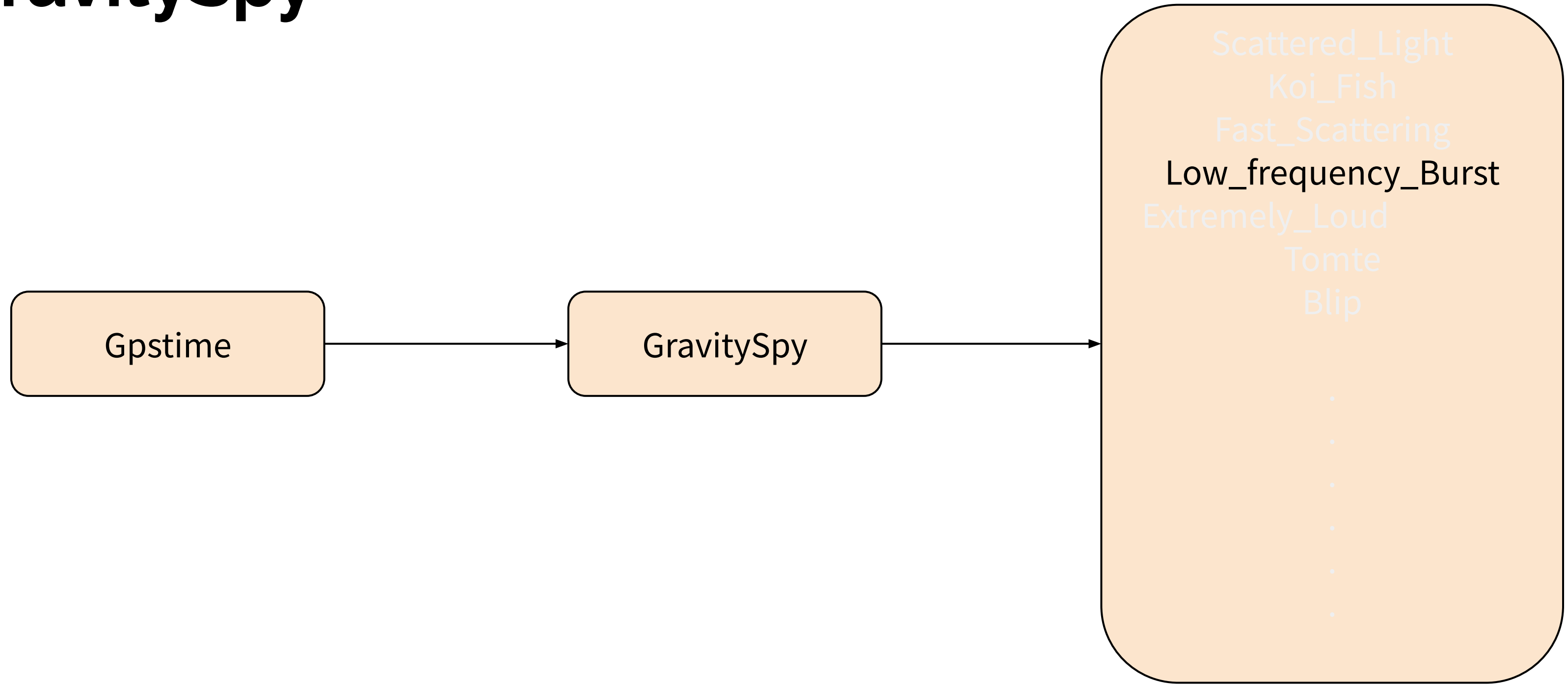
- Transient noise is short bursts of power in the data
- Group transients with similar morphology together
- GravitySpy: a CNN based model that classifies transient noise into separate category based on the time-frequency shape
- As detector evolves new morphologies may show up



GravitySpy

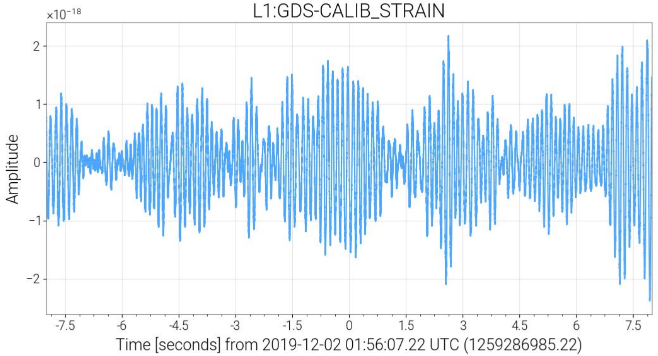

Gpstime
GravitySpy
Scattered_Light
Koi_Fish
Fast_Scattering
Low_frequency_Burst
Extremely_Loud
Tomte
Blip
.
.
.
.
.
.

GravitySpy

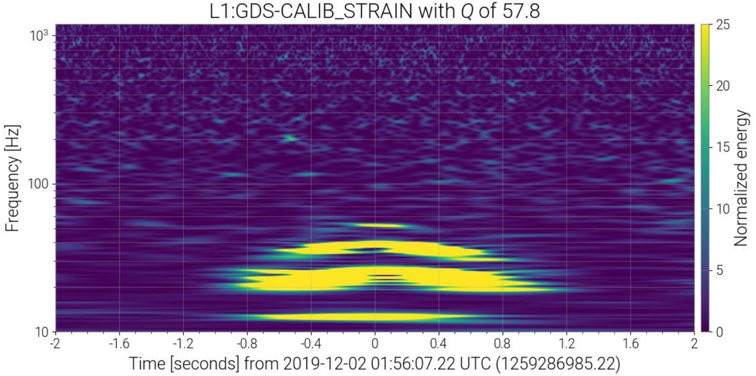

Gpstime
GravitySpy
Scattered_Light
Koi_Fish
Fast_Scattering
Low_frequency_Burst
Extremely_Loud
Tomte
Blip

Training data: high resolution Q-scans

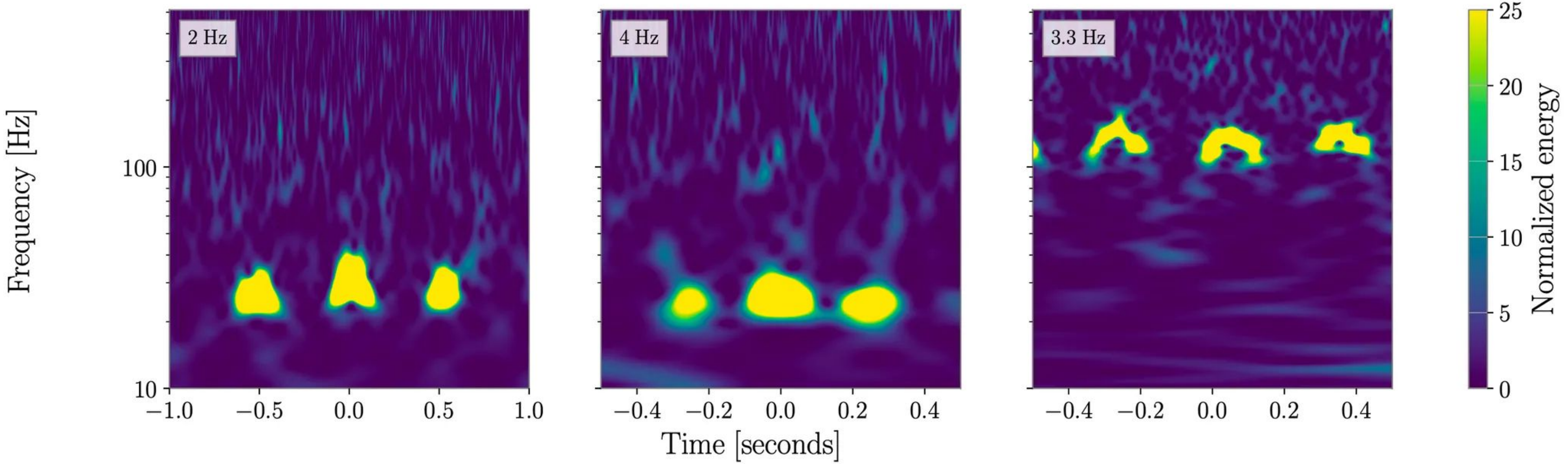
- Start with Omicron triggers (t_0)
- Take the time-series around the gpstime
- Create Q-scans
- Train the model on the Q-scans



What to do when a new transient shape shows up?



Fast Scatter in O3

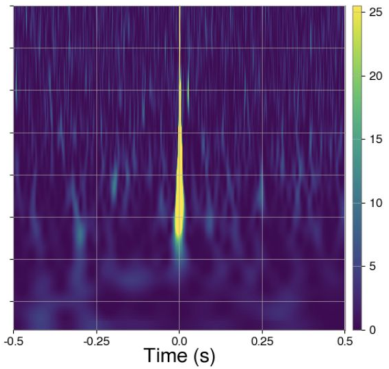


- During the third Observing run, we noticed a new morphology of transient noise in the data
- Correlated with ground motion mostly in 1-5 Hz band
- GravitySpy was misclassifying this noise into other categories
- This misclassification made it difficult to analyze and characterize the noise

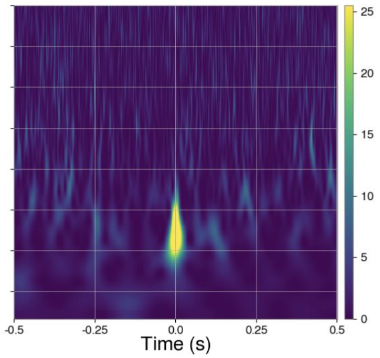
Retraining and Testing

Retraining the model

- New training set: Added Fast Scattering and Low frequency Blips and Removed None_of_the_Above
- Total 9631 training images over 23 classes
- Training and Validation accuracy: 99.9% and 98.8%



Blip



Low f Blip

Old Training Set

[11]:

Blip	1821
Koi_Fish	706
Low_Frequency_Burst	621
Light_Modulation	512
Power_Line	449
Extremely_Loud	447
Low_Frequency_Lines	447
Scattered_Light	443
Violin_Mode	412
Scratchy	337
1080Lines	327
Whistle	299
Helix	279
Repeating_Blips	263
No_Glitch	117
Tomte	103
None_of_the_Above	81
1400Ripples	81
Chirp	60
Air_Compressor	58
Wandering_Line	42
Paired_Doves	27

Name: true_label, dtype: int64

New Training Set

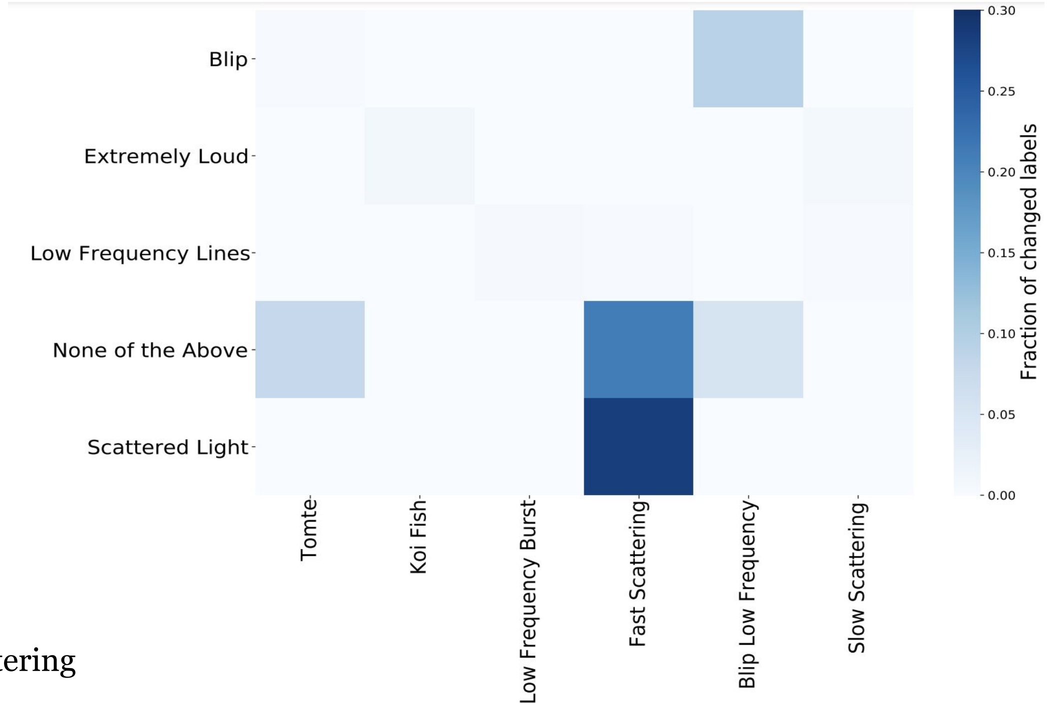
```
df_fastblip2['Label'].value_counts()
```

Blip	1821
Koi_Fish	706
Tomte	703
Blip_Low_Frequency	630
Low_Frequency_Burst	621
Scattered_Light	593
Light_Modulation	512
Power_Line	449
Low_Frequency_Lines	447
Extremely_Loud	447
Violin_Mode	412
Fast_Scattering	400
Scratchy	337
1080Lines	327
Whistle	299
Helix	279
Repeating_Blips	263
No_Glitch	117
1400Ripples	81
Chirp	60
Air_Compressor	58
Wandering_Line	42
Paired_Doves	27

Name: Label, dtype: int64

LLO Heatmap: Old labels to new labels

Old classification



L1 New classification

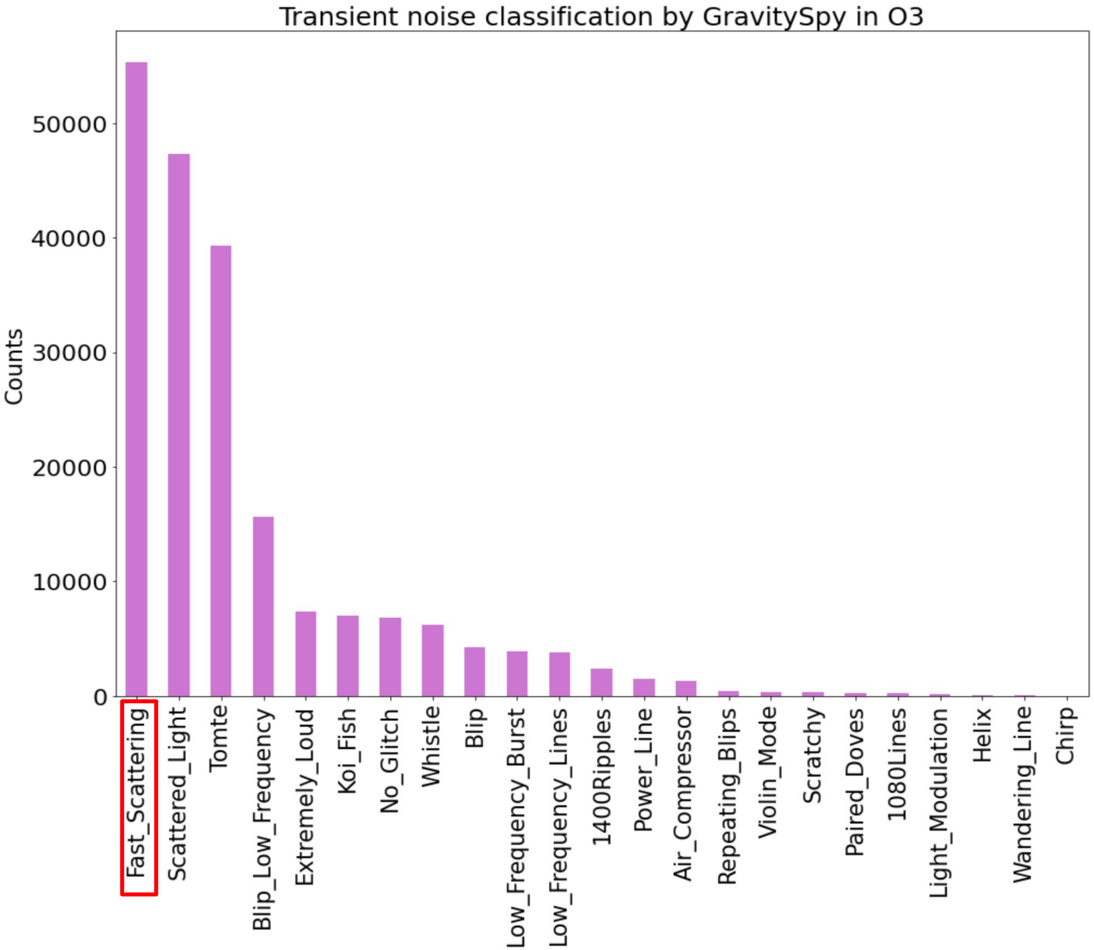
Slow Scattering and None of the above → Fast Scattering

No unexpected change in labels

Fast Scatter in the O3 data

Retrained GravitySpy to identify Fast Scatter transients in the data and then Reclassified the whole O3 dataset

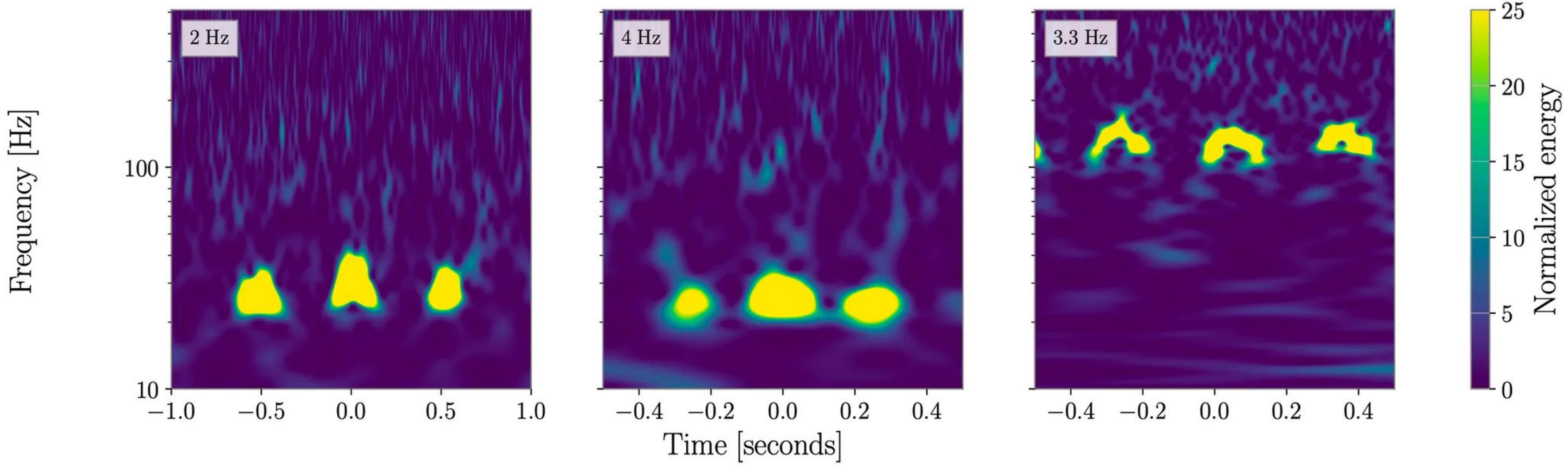
Reclassification showed that Fast Scatter is the most frequent source of transients at LLO in O3



Discovering features in gravitational-wave data through detector characterization, citizen science and machine learning S Soni *et al* 2021 *Class. Quantum Grav.* **38** 195016

Fast Scatter morphology

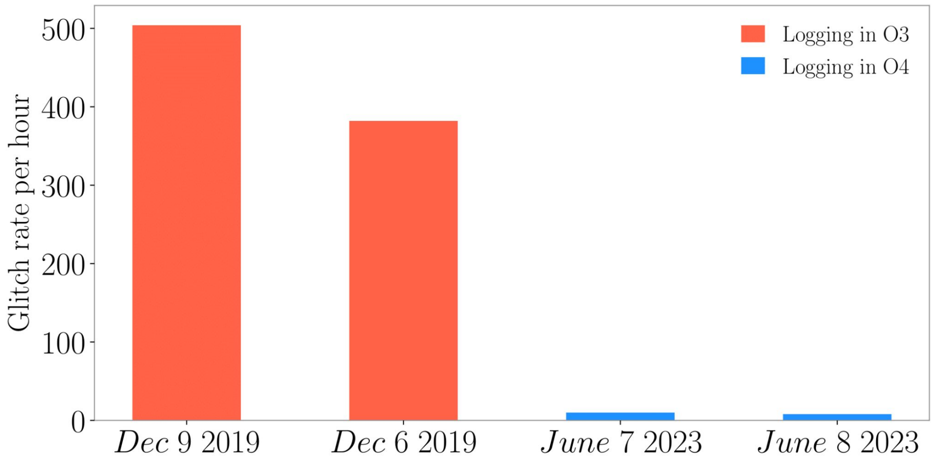
- ML assisted noise identification → Two major population (2 and 4 Hz) of fast scatter in the data



2 Hz and 4 Hz during O3, 3.3 Hz in the Post O3 data

GravitySpy Retraining Impact

- Retraining and reclassification allowed improved characterization of the Fast Scatter noise
- This led to improved noise modeling which in turn led us to the source
- Once source was fixed, the noise is nearly eliminated from the data
- Example of how machine learning can assist in noise modeling and noise reduction



Modeling and Reduction of high frequency scatter at LIGO Livingston Soni *et al* 2024 *Class. Quantum Grav.* **41** 135015

An analysis of O4a glitches using t-SNE

Machine Learning

Identification of glitch groups

Omicron information

Correlation with physical or instrumentation conditions

An Analysis of LIGO Glitches Using t-SNE During the First Part of the Fourth LIGO-Virgo-KAGRA Observing Run

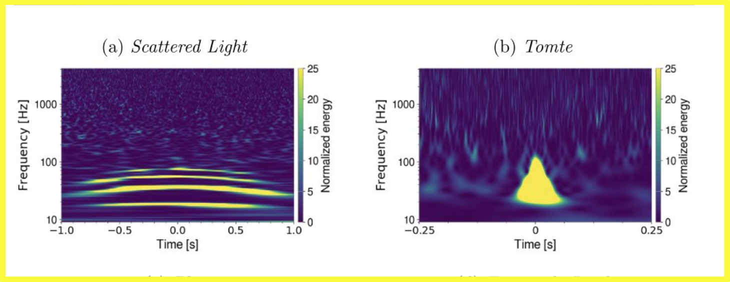
Tabata Aira Ferreira, Gabriela González, Osvaldo Salas

This paper presents an analysis of noise transients observed in LIGO data during the first part of the fourth observing run, using the unsupervised machine learning technique t-distributed Stochastic Neighbor Embedding (t-SNE) to examine the behavior of glitch groups. Based on the t-SNE output, we apply Agglomerative Clustering in combination with the Silhouette Score to determine the optimal number of groups. We then track these groups over time and investigate correlations between their occurrence and environmental or instrumental conditions. At the Livingston observatory, the most common glitches during O4a were seasonal and associated with ground motion, whereas at Hanford, the most prevalent glitches were related to instrumental conditions.

Machine learning enables computers to learn from data, make predictions, and identify patterns.

Fokus AI, Credit: Phonlamai Photo/Shutterstock

Supervised



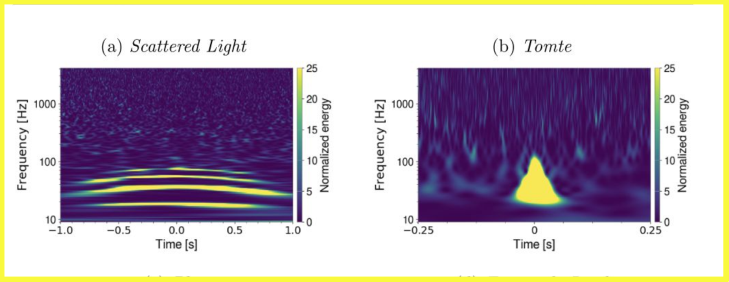
Adapted from Ferreira et al. (2024)

- Requires *previous knowledge* of the data to train a model for classifying new glitches

- Works very well and provides a *confidence level* for each new classification

- Performance depends on the training dataset

Supervised



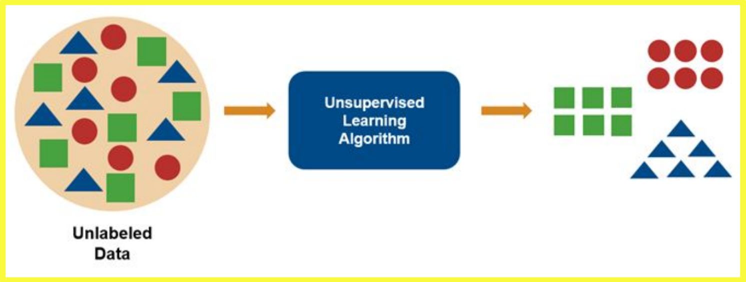
Adapted from Ferreira et al. (2024)

- Requires *previous knowledge* of the data to train a model for classifying new glitches

- Works very well and provides a *confidence level* for each new classification

- Performance depends on the training dataset

Unsupervised



https://www.mathworks.com/discovery/unsupervised-learning.html

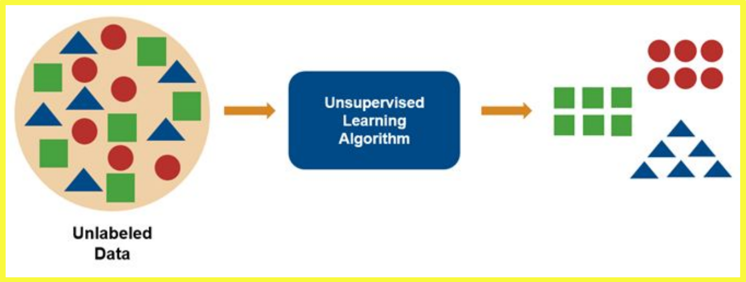
- Does not require *previous knowledge* of the data to train a model for classifying new glitches

- In our case, there is no risk of misclassification, since the method does not rely on predefined classes

Unsupervised

t-SNE (t-Distributed Stochastic Neighbor Embedding)

- visualization technique (dimensionality reduction)
- Projects high-dimensional data → 2D/3D
- Preserves similarity between points
- Reveals clusters & structures

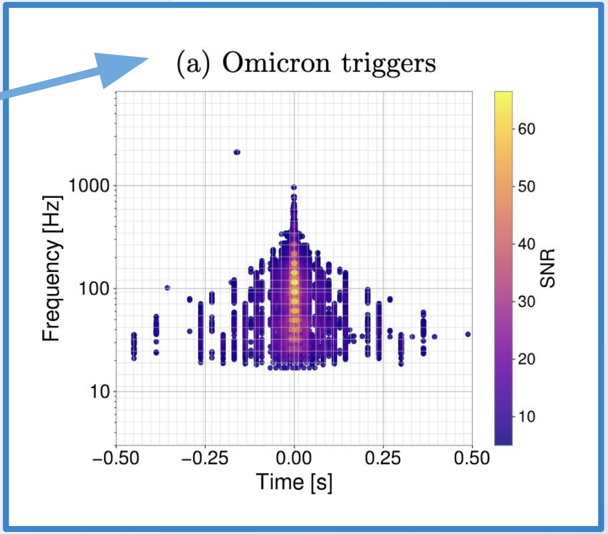


https://www.mathworks.com/discovery/unsupervised-learning.html

- Does not require *previous knowledge* of the data to train a model for classifying new glitches
- In our case, there is no risk of misclassification, since the method does not rely on predefined classes

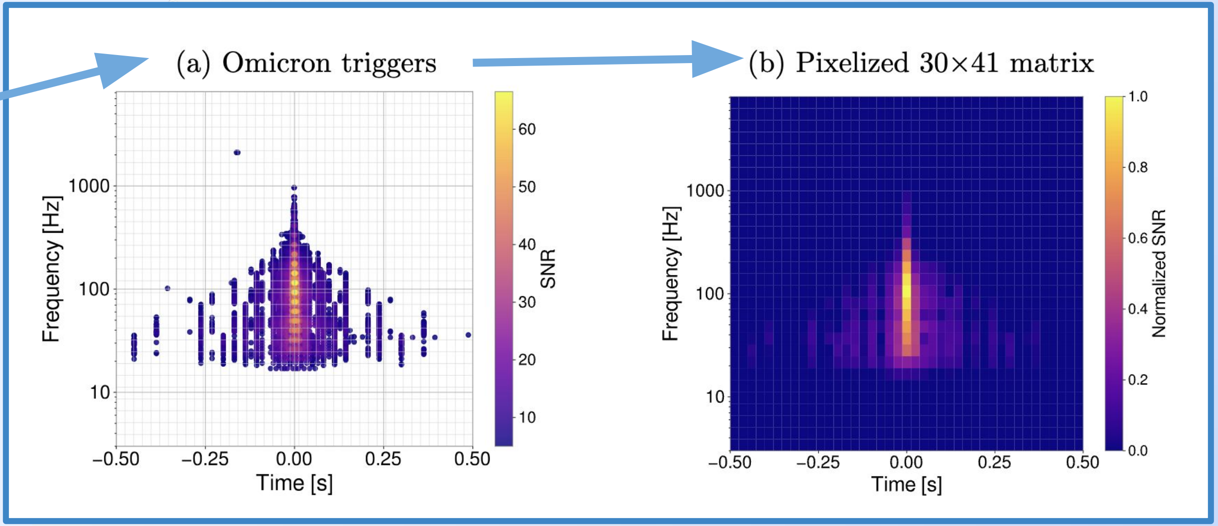
Omicron creates unclustered information different Q-values

List of glitch times



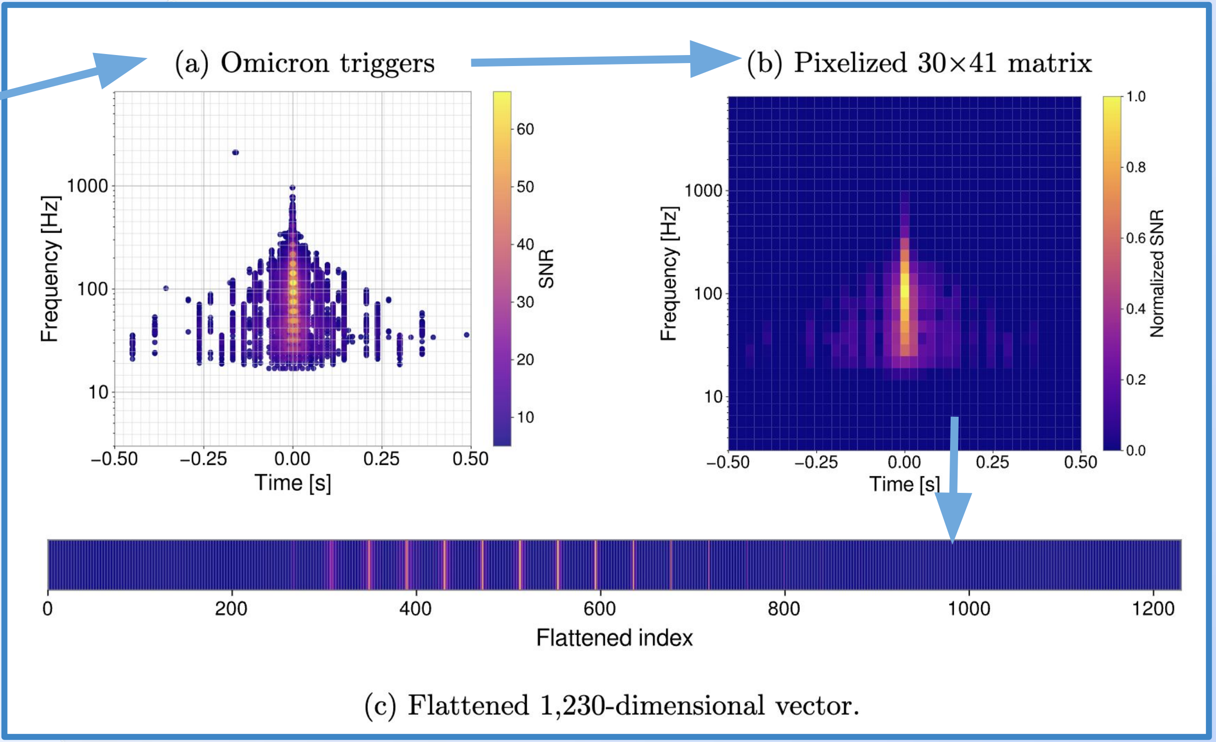
Omicron creates unclustered information different Q-values

List of glitch times



Omicron creates unclustered information different Q-values

List of glitch times

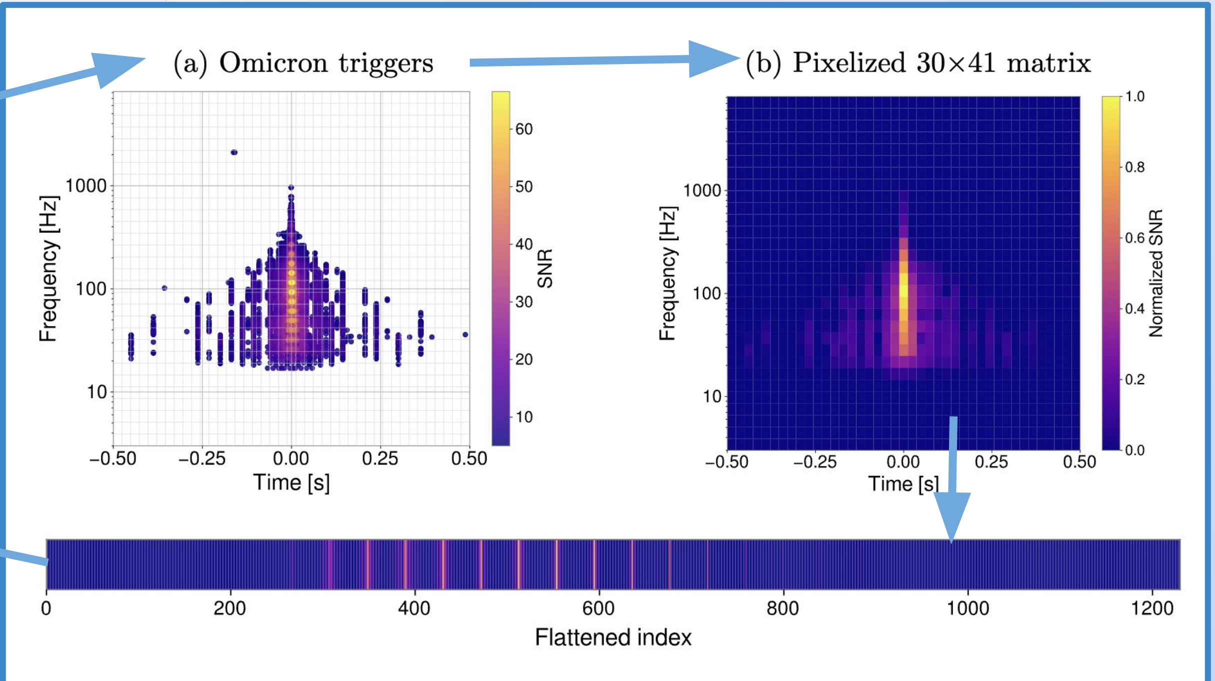


(c) Flattened 1,230-dimensional vector.

Omicron creates unclustered information different Q-values

List of glitch times

A data vector representing one glitch

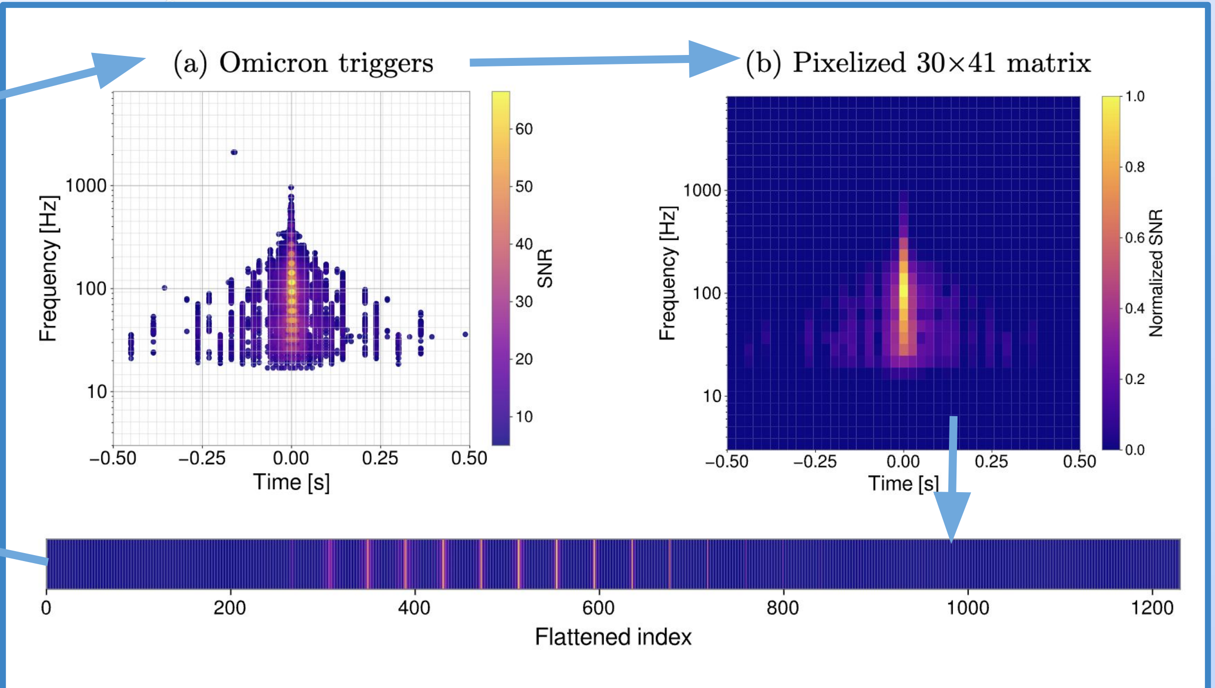


(c) Flattened 1,230-dimensional vector.

Omicron creates unclustered information different Q-values

List of glitch times

A data vector representing one glitch



(c) Flattened 1,230-dimensional vector.

Thousands of vectors with 1,230 dimensions compose our input dataset for t-SNE.

t-SNE reduces the dimensionality of each vector to a lower space (2D or 3D), allowing visualization and producing two or three coordinates as output.

Vinput = [a1, a2, a3, a4, a5, ..., a1230]

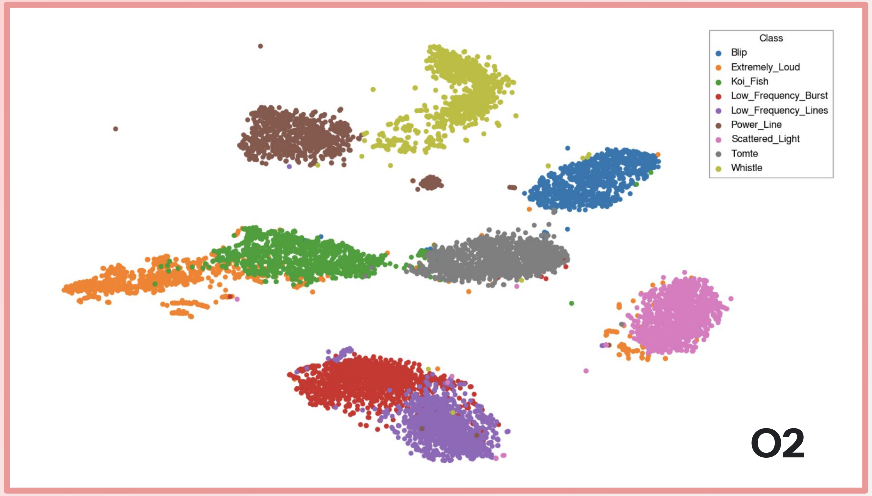
Voutput = [o1, o2]

Validation

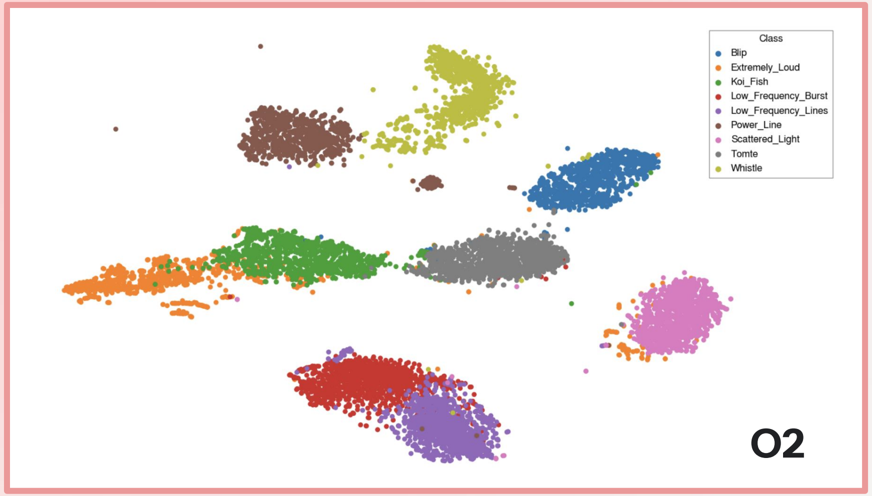
Scatter plot of the two coordinates produced by t-SNE

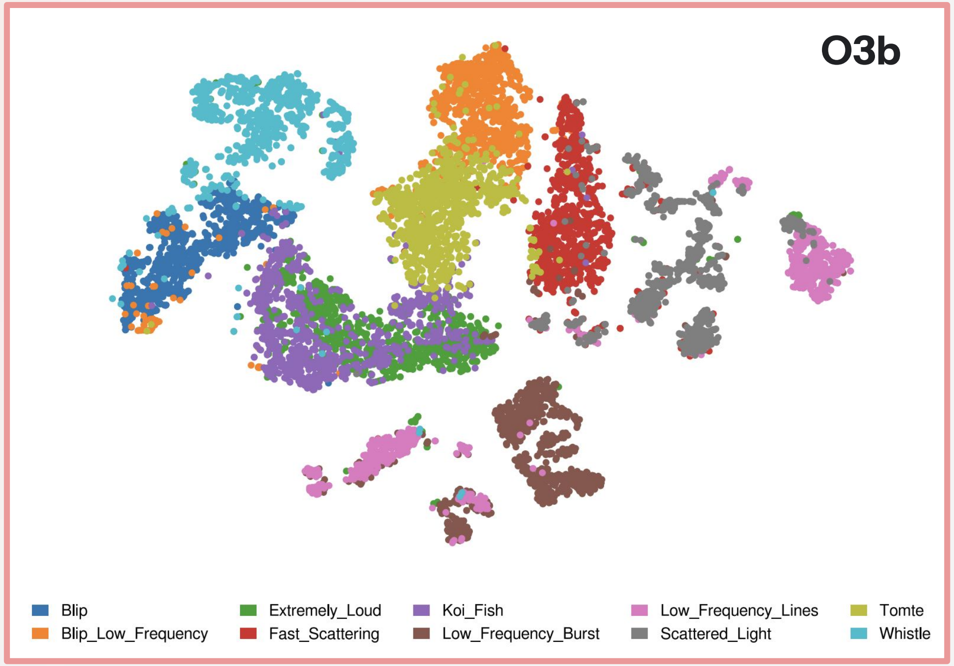
Validation

Coloring according to Gravity Spy

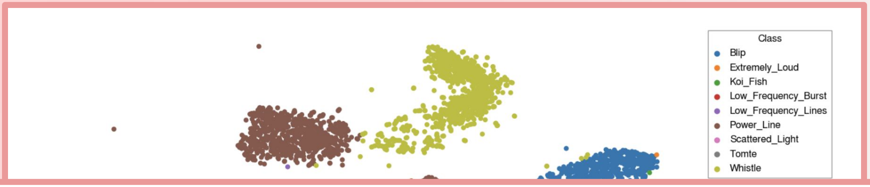


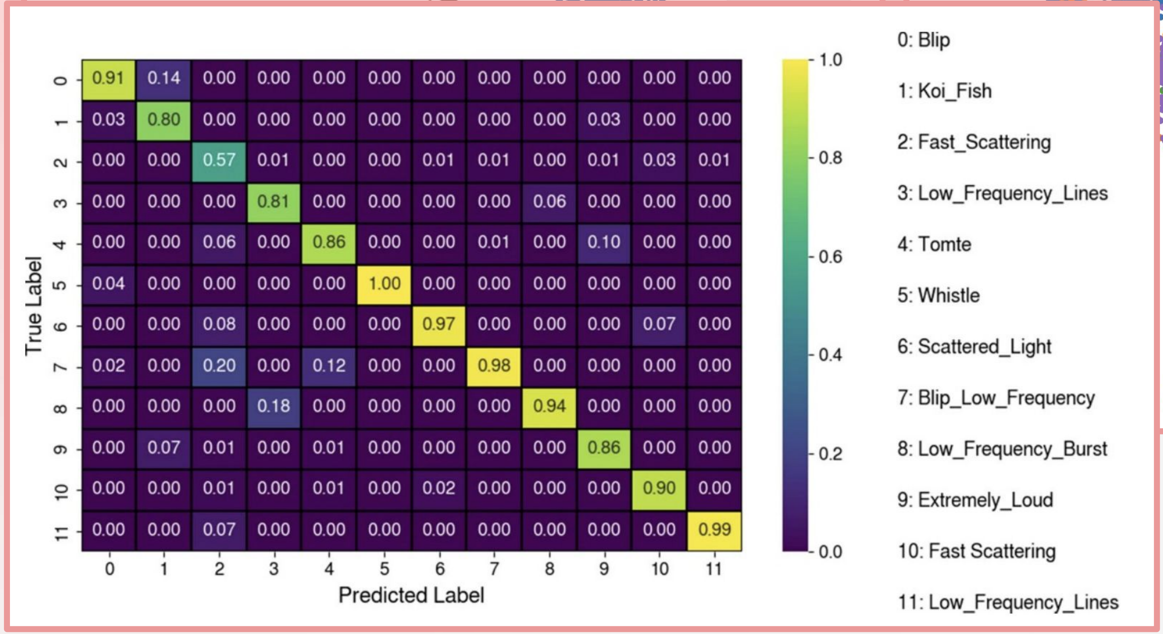
Validation

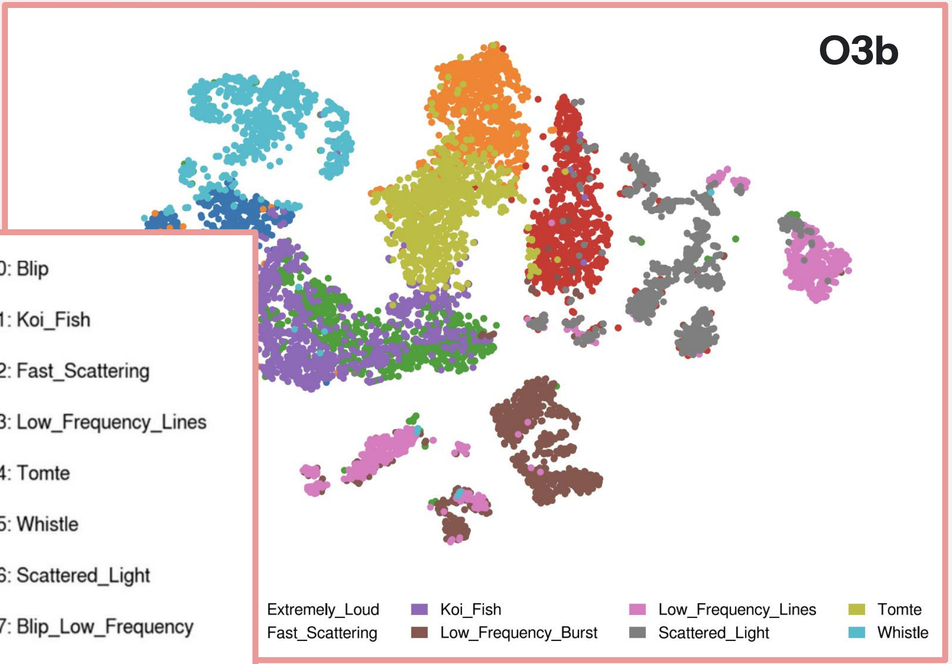

Class
Blip
Extremely_Loud
Koi_Fish
Low_Frequency_Burst
Low_Frequency_Lines
Power_Line
Scattered_Light
Tomte
Whistle
O2


O3b
Blip
Blip_Low_Frequency
Extremely_Loud
Fast_Scattering
Koi_Fish
Low_Frequency_Burst
Low_Frequency_Lines
Scattered_Light
Tomte
Whistle

Validation


Class
Blip
Extremely_Loud
Koi_Fish
Low_Frequency_Burst
Low_Frequency_Lines
Power_Line
Scattered_Light
Tomte
Whistle

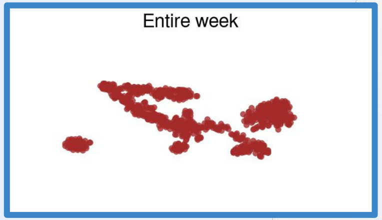

0 | 0.91 0.14 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1 | 0.03 0.80 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.03 0.00 0.00
2 | 0.00 0.00 0.57 0.01 0.00 0.00 0.01 0.01 0.00 0.01 0.03 0.01
3 | 0.00 0.00 0.00 0.81 0.00 0.00 0.00 0.00 0.06 0.00 0.00 0.00
4 | 0.00 0.00 0.06 0.00 0.86 0.00 0.00 0.01 0.00 0.10 0.00 0.00
5 | 0.04 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
6 | 0.00 0.00 0.08 0.00 0.00 0.00 0.97 0.00 0.00 0.00 0.07 0.00
7 | 0.02 0.00 0.20 0.00 0.12 0.00 0.00 0.98 0.00 0.00 0.00 0.00
8 | 0.00 0.00 0.00 0.18 0.00 0.00 0.00 0.00 0.94 0.00 0.00 0.00
9 | 0.00 0.07 0.01 0.00 0.01 0.00 0.00 0.00 0.00 0.86 0.00 0.00
10 | 0.00 0.00 0.01 0.00 0.01 0.00 0.02 0.00 0.00 0.00 0.90 0.00
11 | 0.00 0.00 0.07 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.99
True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9 10 11
1.0 0.8 0.6 0.4 0.2 0.0
0: Blip
1: Koi_Fish
2: Fast_Scattering
3: Low_Frequency_Lines
4: Tomte
5: Whistle
6: Scattered_Light
7: Blip_Low_Frequency
8: Low_Frequency_Burst
9: Extremely_Loud
10: Fast Scattering
11: Low_Frequency_Lines


O3b
Extremely_Loud
Fast_Scattering
Koi_Fish
Low_Frequency_Burst
Low_Frequency_Lines
Scattered_Light
Tomte
Whistle

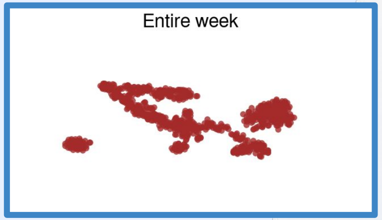
We can characterize glitch classes in this way and identify them in the gravitational-wave channel, but what can we practically achieve with this method?

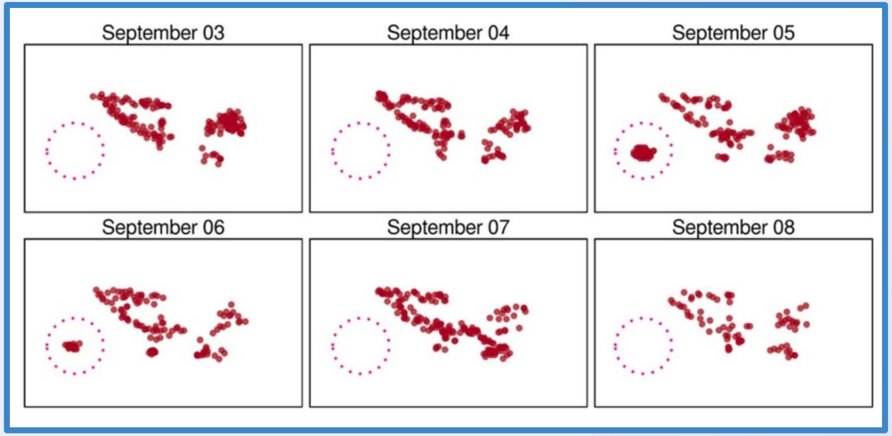
Application Examples (LLO)

Looking for outliers throughout a week

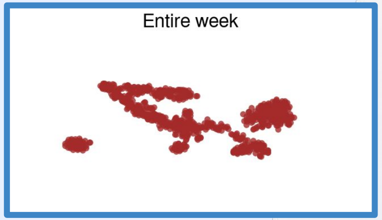


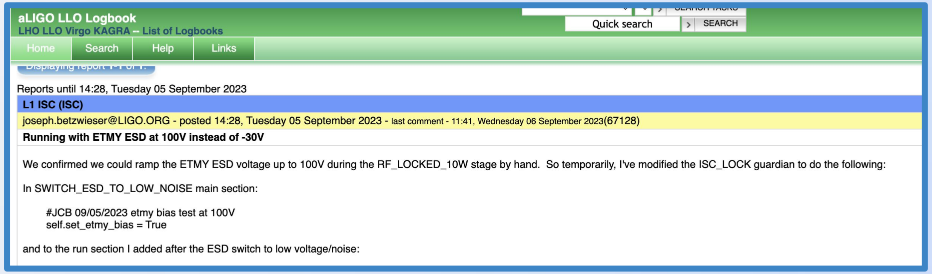
Looking for outliers throughout a week

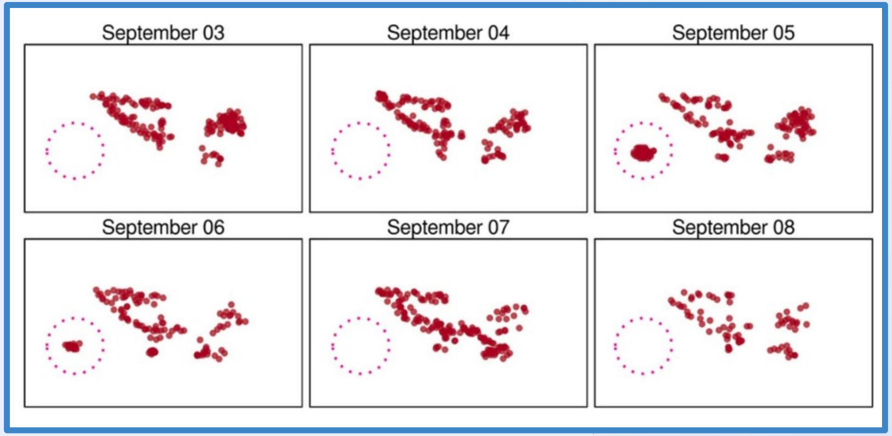




Looking for outliers throughout a week

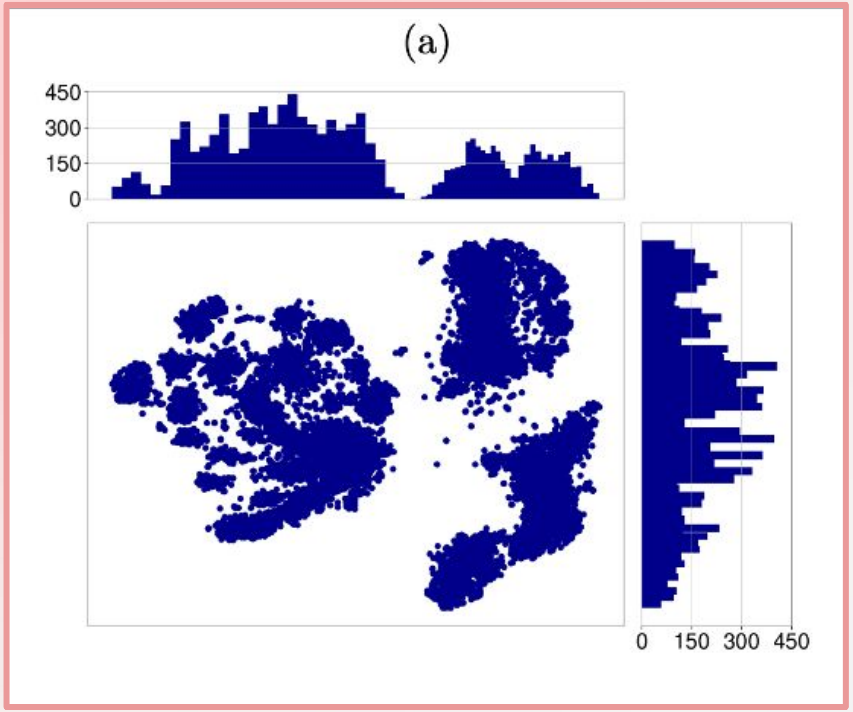




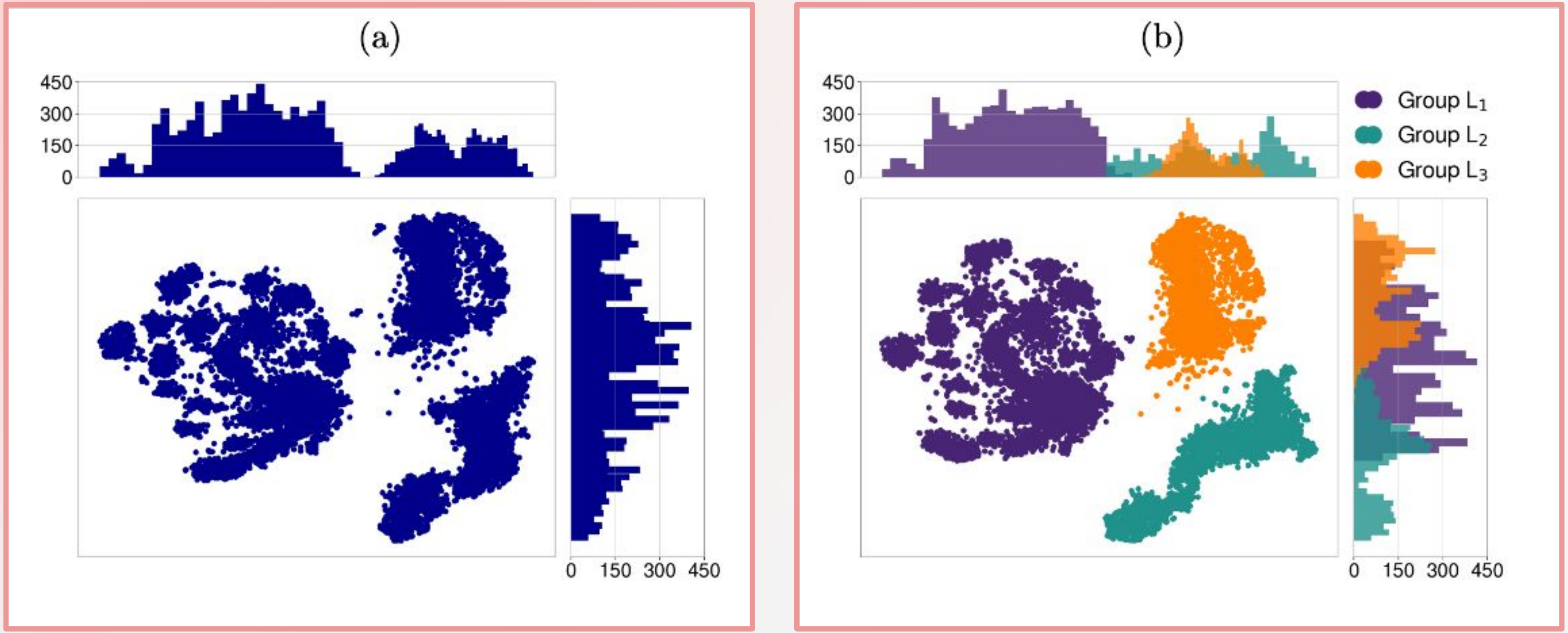


The glitches in the outlier group were related to instrumentation issues

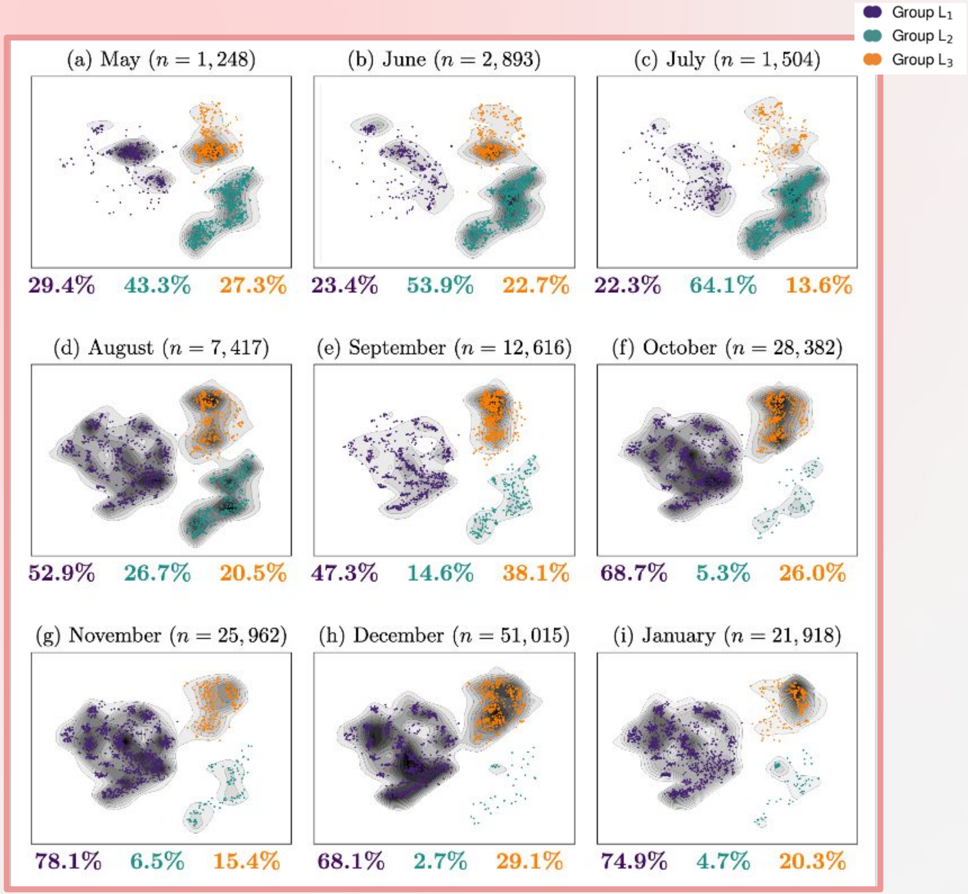
Month-by-month analysis during O4a



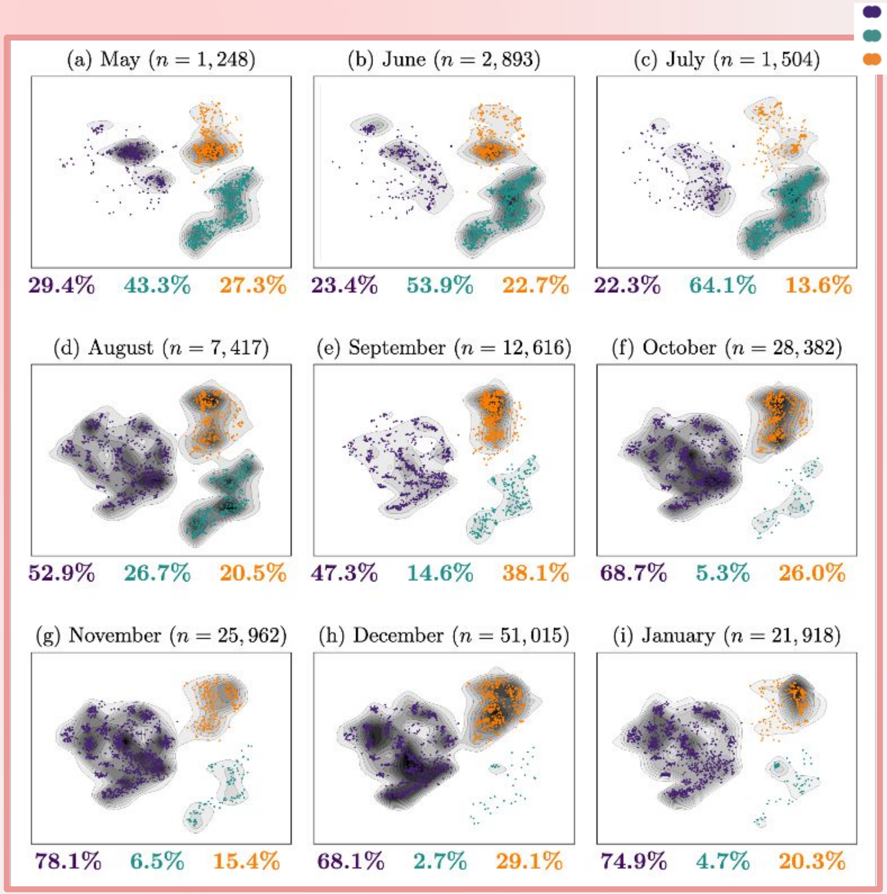
Month-by-month analysis during O4a

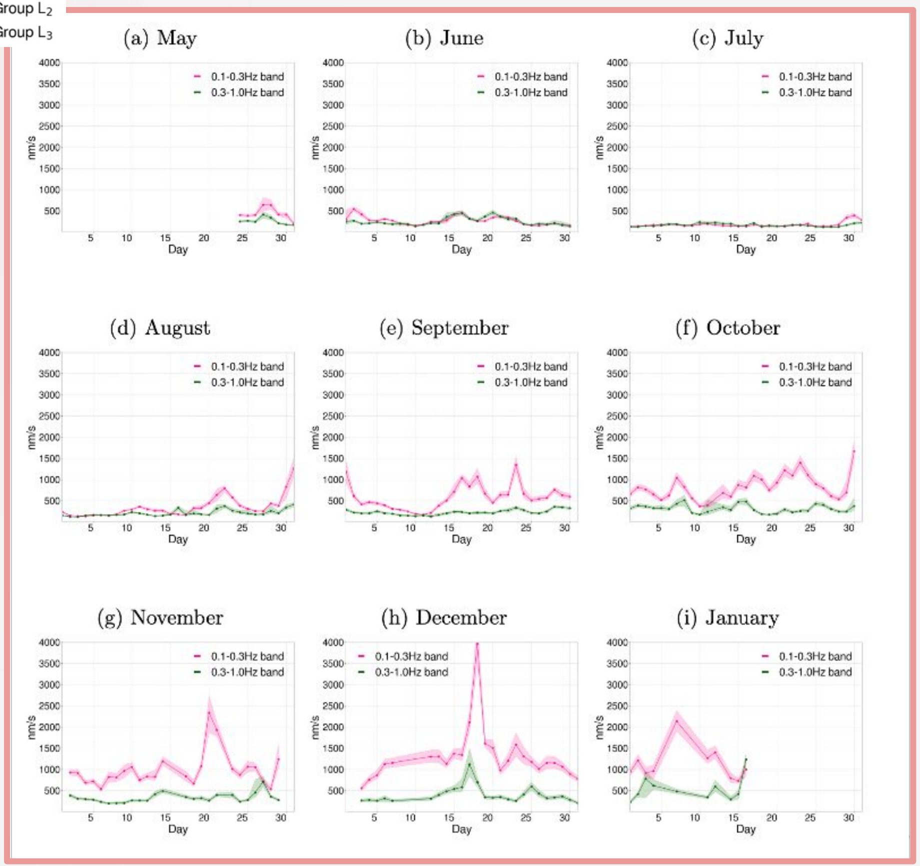


Month-by-month analysis during O4a

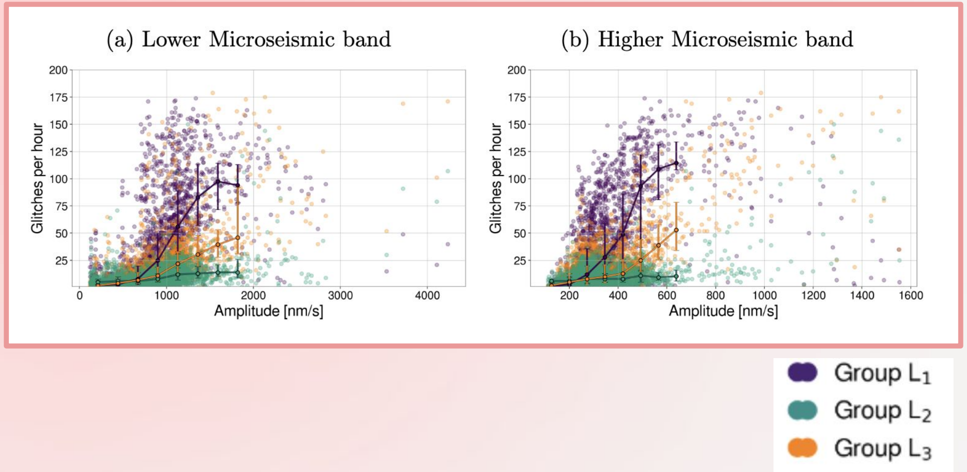


Month-by-month analysis during O4a



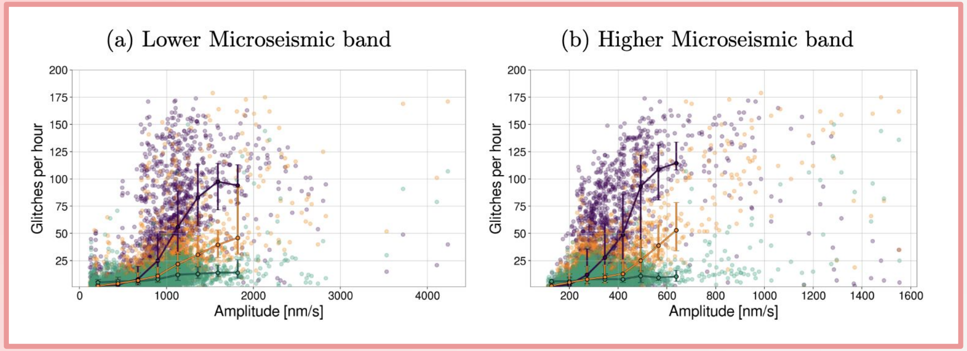


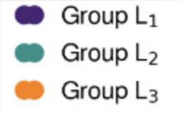
Month-by-month analysis during O4a

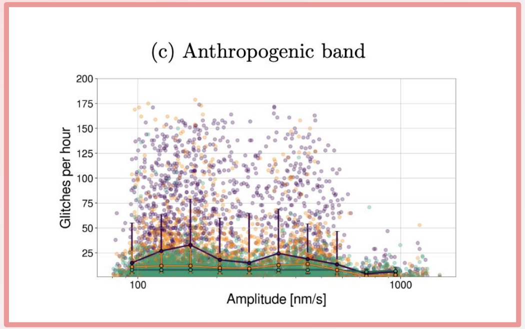


Confirming the correlation between the groups identified by the method and ground motion in different frequency bands...

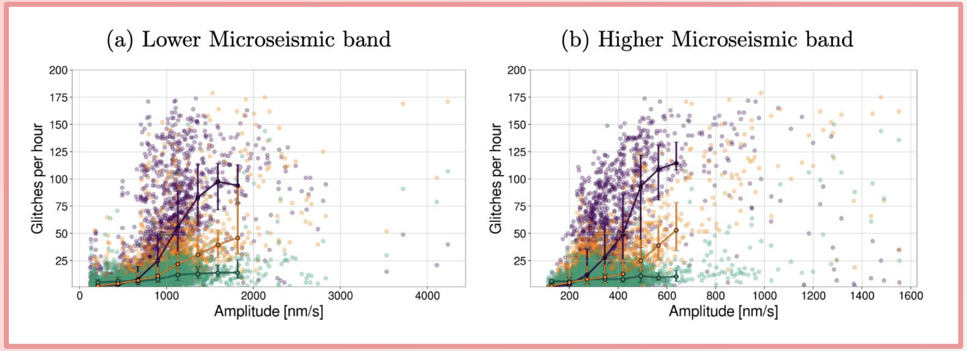
Month-by-month analysis during O4a

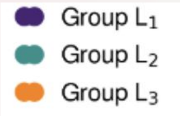


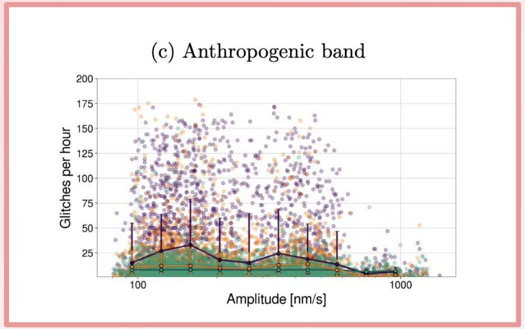




Month-by-month analysis during O4a

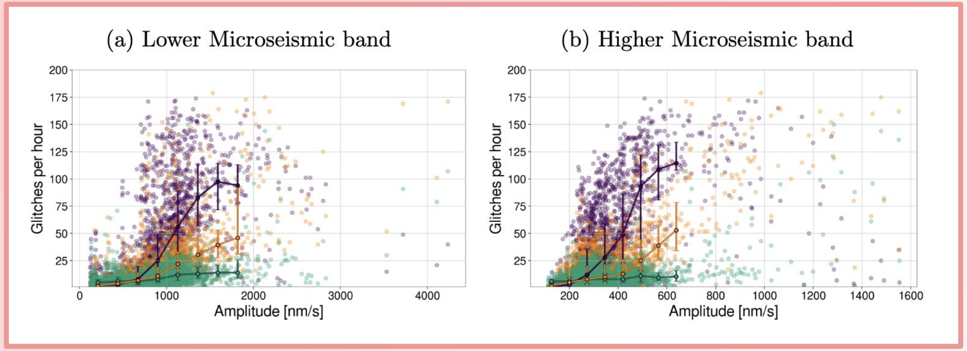




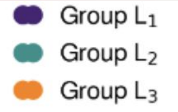


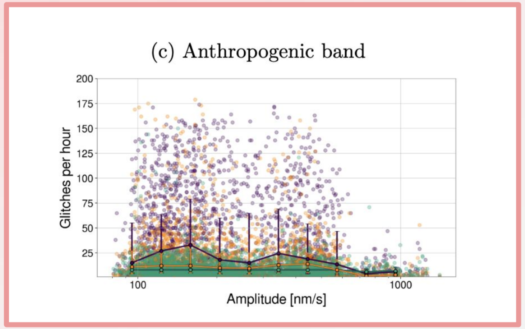
Correlation	LMS	HMS	Ant. band
Group L_1	0.64	0.63	-0.03
Group L_2	0.32	0.17	-0.04
Group L_3	0.53	0.47	-0.05

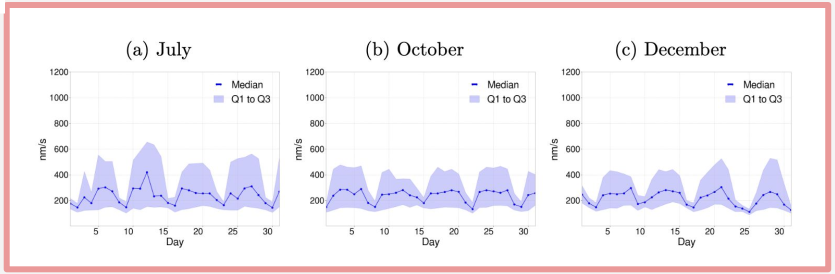
Month-by-month analysis during O4a



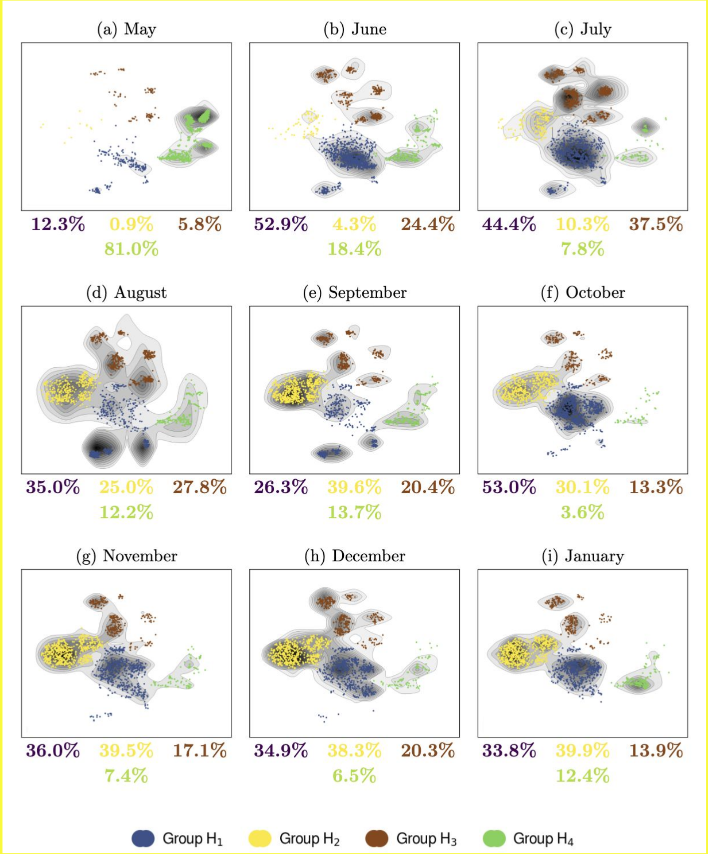
Correlation	LMS	HMS	Ant. band
Group L_1	0.64	0.63	-0.03
Group L_2	0.32	0.17	-0.04
Group L_3	0.53	0.47	-0.05

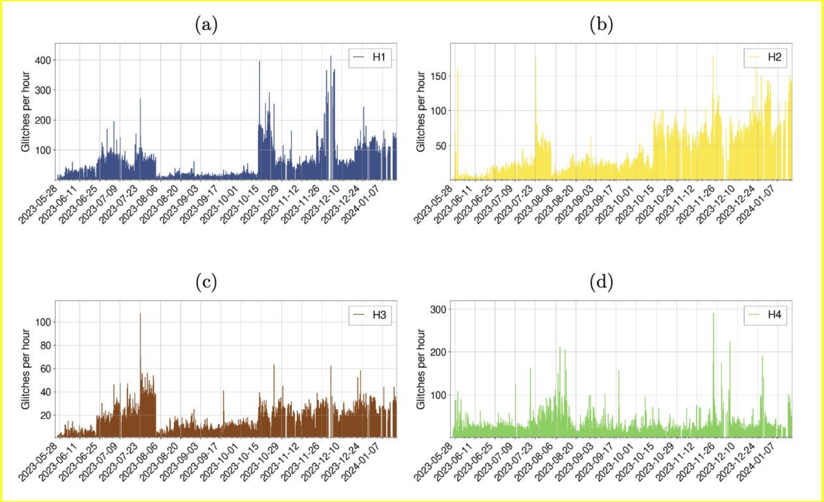


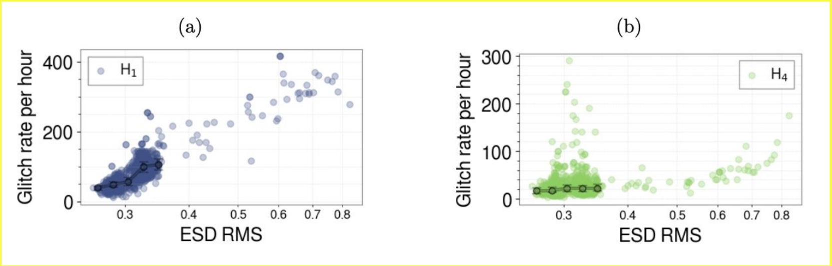




Month-by-month analysis during O4a (LHO)







Related to nonlinear couplings of the ESD actuation system with the detector signal.

Pros

- Bases only on Omicron - no need for image - quick analysis.
- Identifies groups of similar glitches and tracks them over time at different scales (seconds, days, weeks, months).
- Provides the flexibility to use different Q ranges, which may vary depending on the goal of the analysis.
- Low latency (depending on Omicron).
- No need for a training dataset.
- Can be used to identify groups as outliers.
- Can be used regardless of labeled classes.
- The classifier is fully automated and independent of user input or the number of clusters.
- Can be applied to **auxiliary channels.**

Cons

- Does not work without Omicron.
- Since there is no need for a training dataset, its main goal is not to be a classifier.
- Class identification is biased by the t-SNE output.
- If Omicron does not run on a given auxiliary channel, this analysis is not possible.

Thank you!

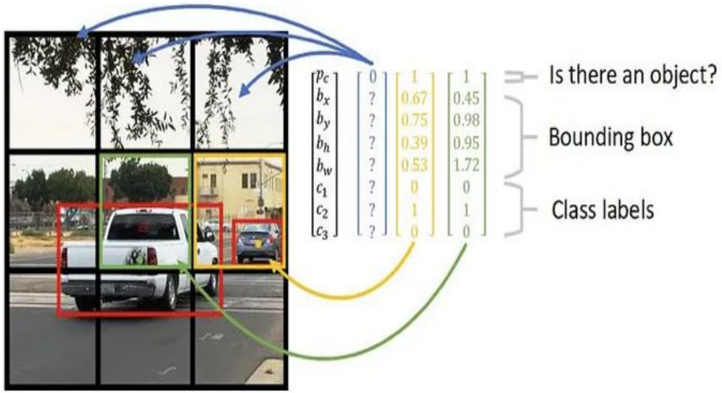
GW-YOLO

Computer Vision

- Field within AI, enabling computers to interpret visual information
- Object recognition, image classification, feature detection, image (pixel) segmentation, tracking motion
- Dominated by Convolutional Neural Networks from 1980-2015
- You Only Look Once (YOLO) since 2016 being used for object detection in real time

You Only Look Once aka YOLO

- YOLO is a family of real-time object detection algorithms.
- Input → YOLO → (bounding box/masks, class label) single shot detection
- Divides the image into an NxN grid and for each grid cell it predicts:
  - Object detection: Object exists or not
  - Localization: Bounding box parameters (x,y)
  - Confidence: Class probability
- Really fast: 150 fps, great for real time prediction.
- Applications:
  - Self-driving cars
  - Medical imaging
  - Industrial defect detection and so much more



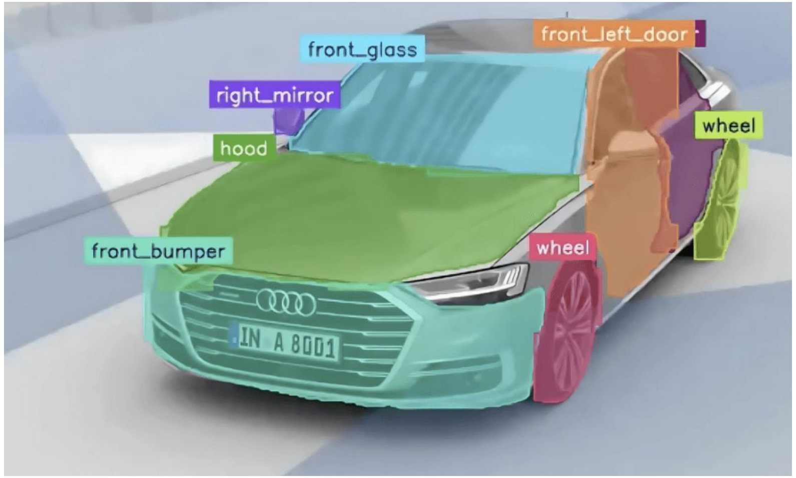
YOLO algorithm
https://arxiv.org/abs/1506.02640

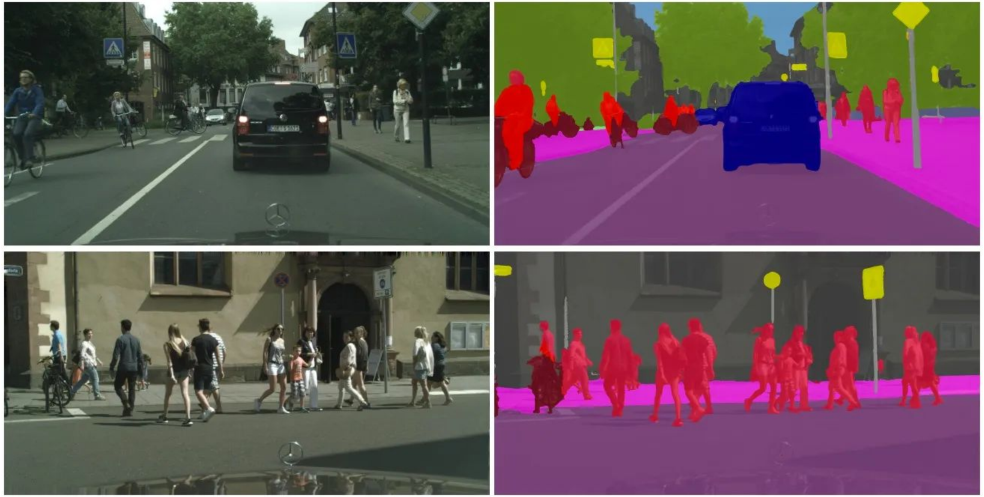
Image source

Image Segmentation

- Beyond object detection or classification
- **Pixel level masks** that outline the shape of the object
- Instance segmentation: ***what*** the object is and ***where*** it is
- Multiclass image segmentation
- Used in autonomous driving

Image source, source

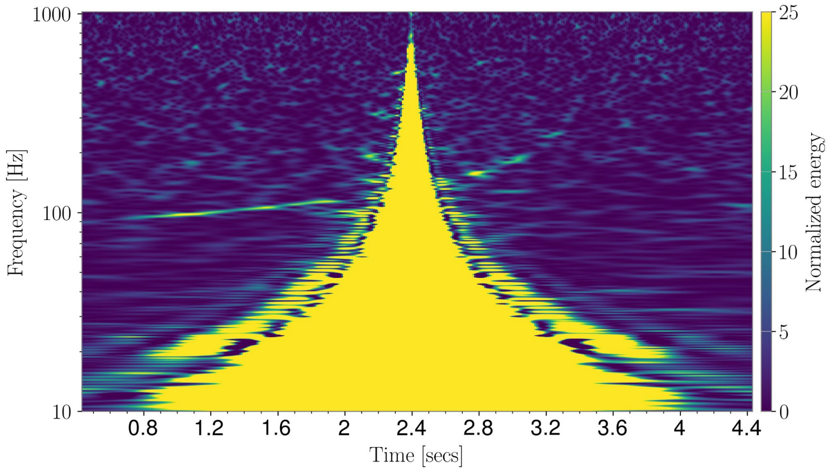


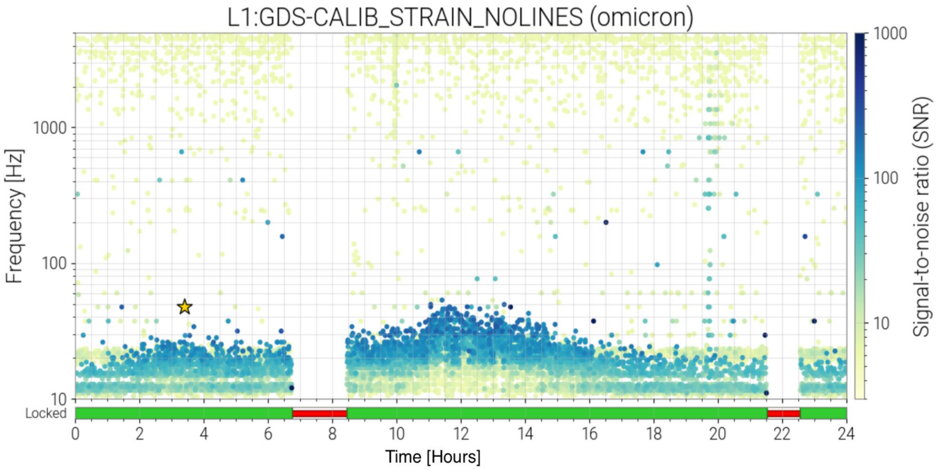


Computer Vision in LIGO

Motivation

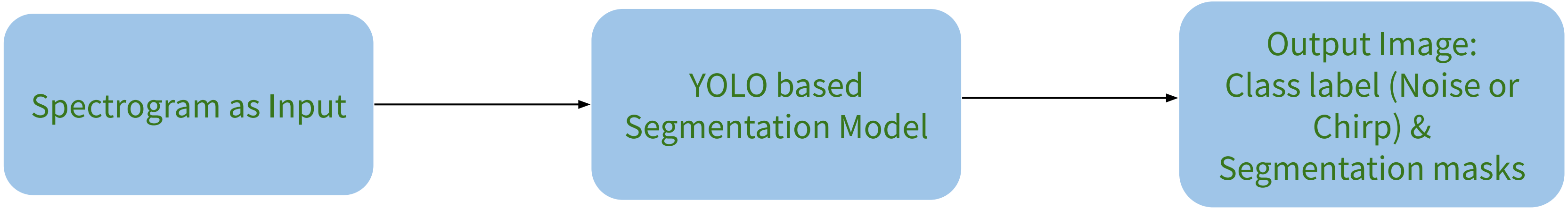
GW170817





As detectors become more sensitive, the rate of transient noise may go up

Main Idea



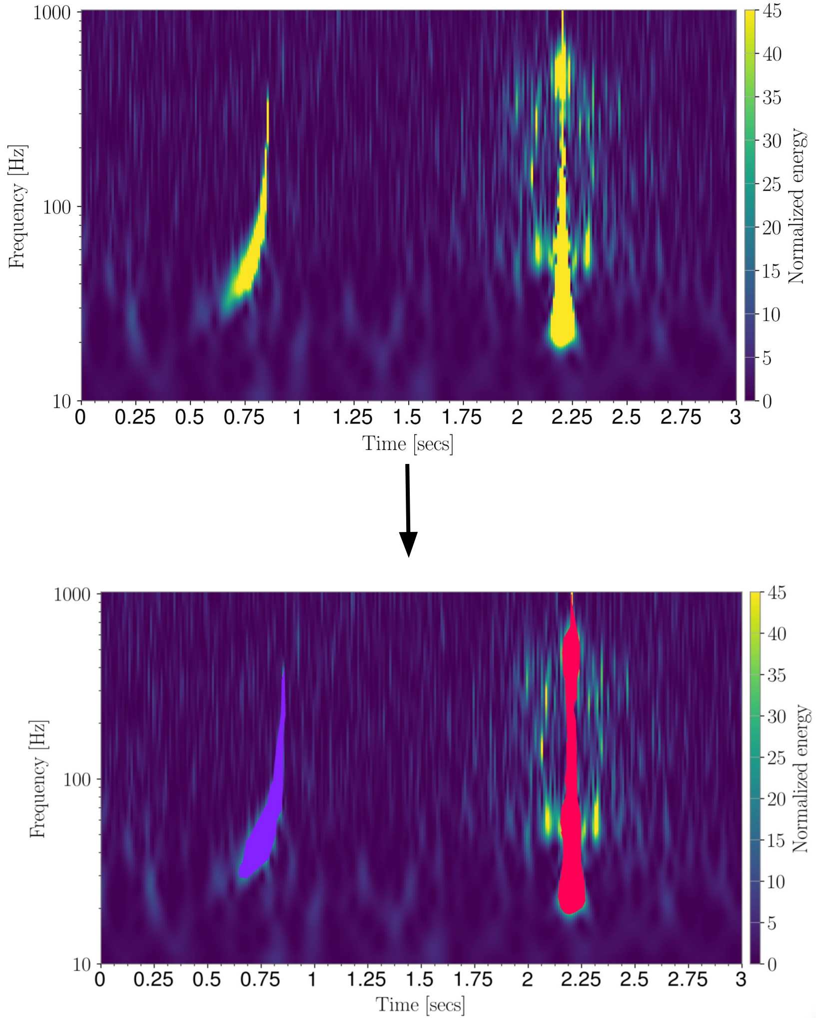
We feed Q transforms (or event times) to the script and it outputs a Q scan showing different class labels and segmented pixel-masks for each class

For this, we first need to train our model on lots of annotated spectrograms of transient noise and chirps.

Training data

- Noise: Glitches from O3
- Signals: BBH , BNS signal waveforms, generated using PyCBC and O3 injection dataset
- Combine the glitches and chirps, make the Q-transform, annotate the chirps and noise
- Variation in chirp strength, types of glitches, temporal separation (including overlap) between glitch and chirp in the training set

1. https://zenodo.org/records/5649212
2. https://zenodo.org/records/7890437



Training Metrics

- Metrics to quantify training performance
- **MAP@50:** Mean Avg precision @50 means prediction is correct if Intersection Over Union area between predicted and ground truth is > 50. So better localization.
- **Precision**: TP/(TP + FP). High precision leads to fewer wrong guess.
- **Recall**: TP/(TP + FN). High recall means smaller number of missed detections.
- Next: Inference on new examples and then a larger statistical study

DataSet	mAP50	Precision	Recall
Validation	94.7 %	89.0 %	90.0 %
Test	95.3 %	91.5 %	92.0 %

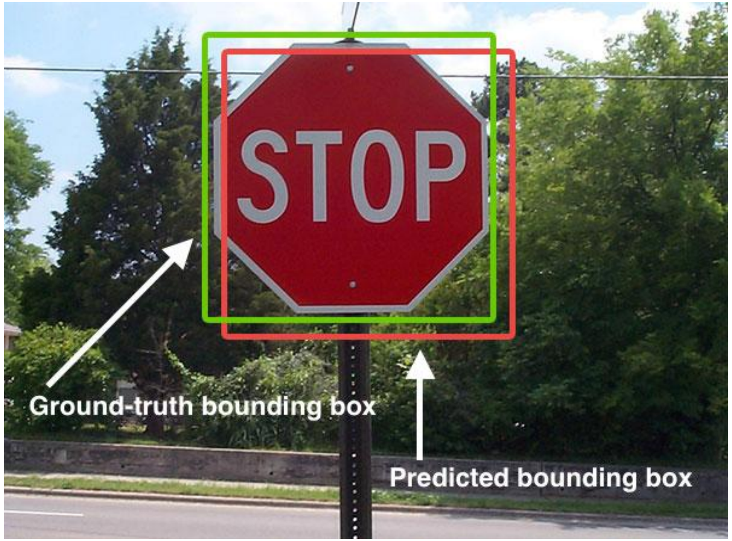
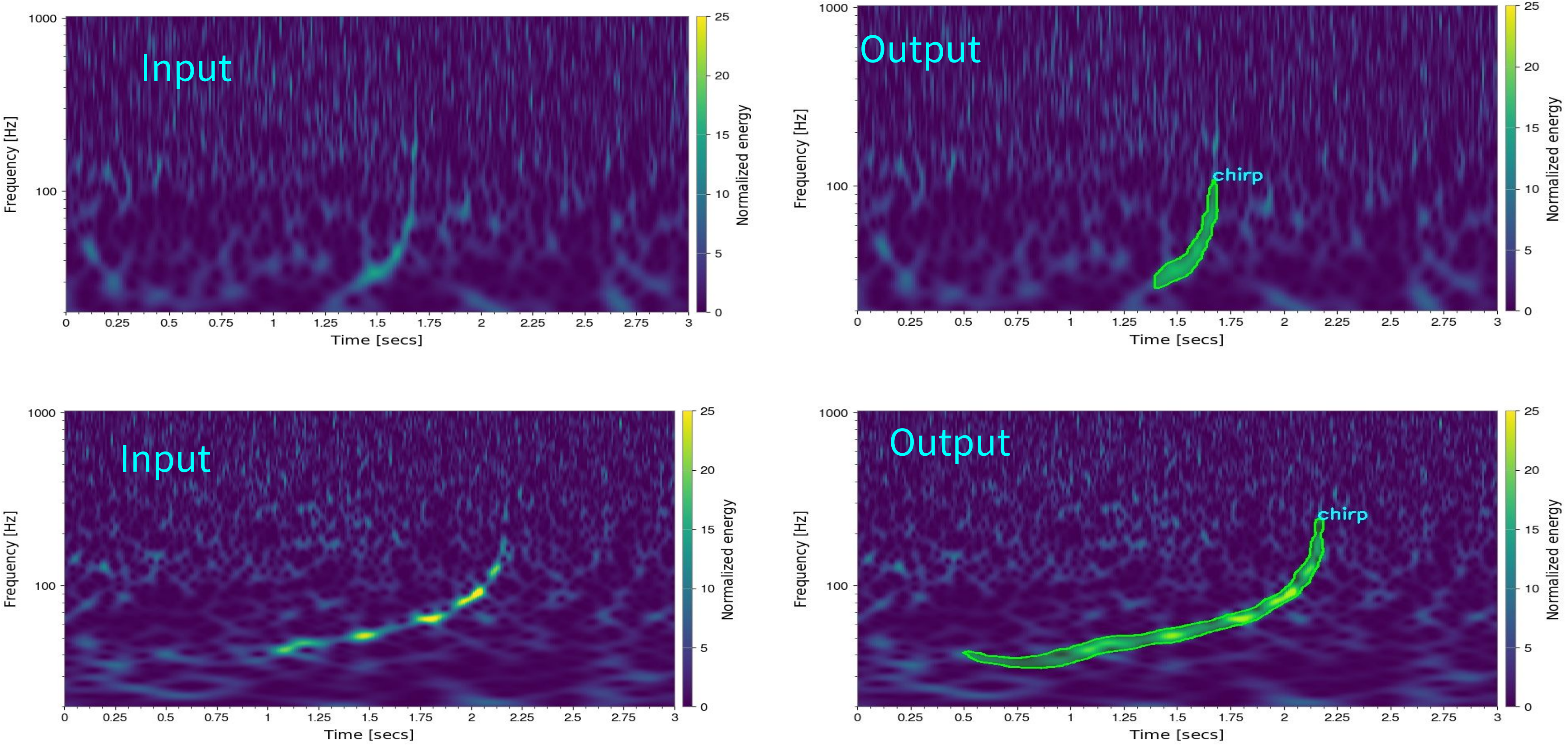


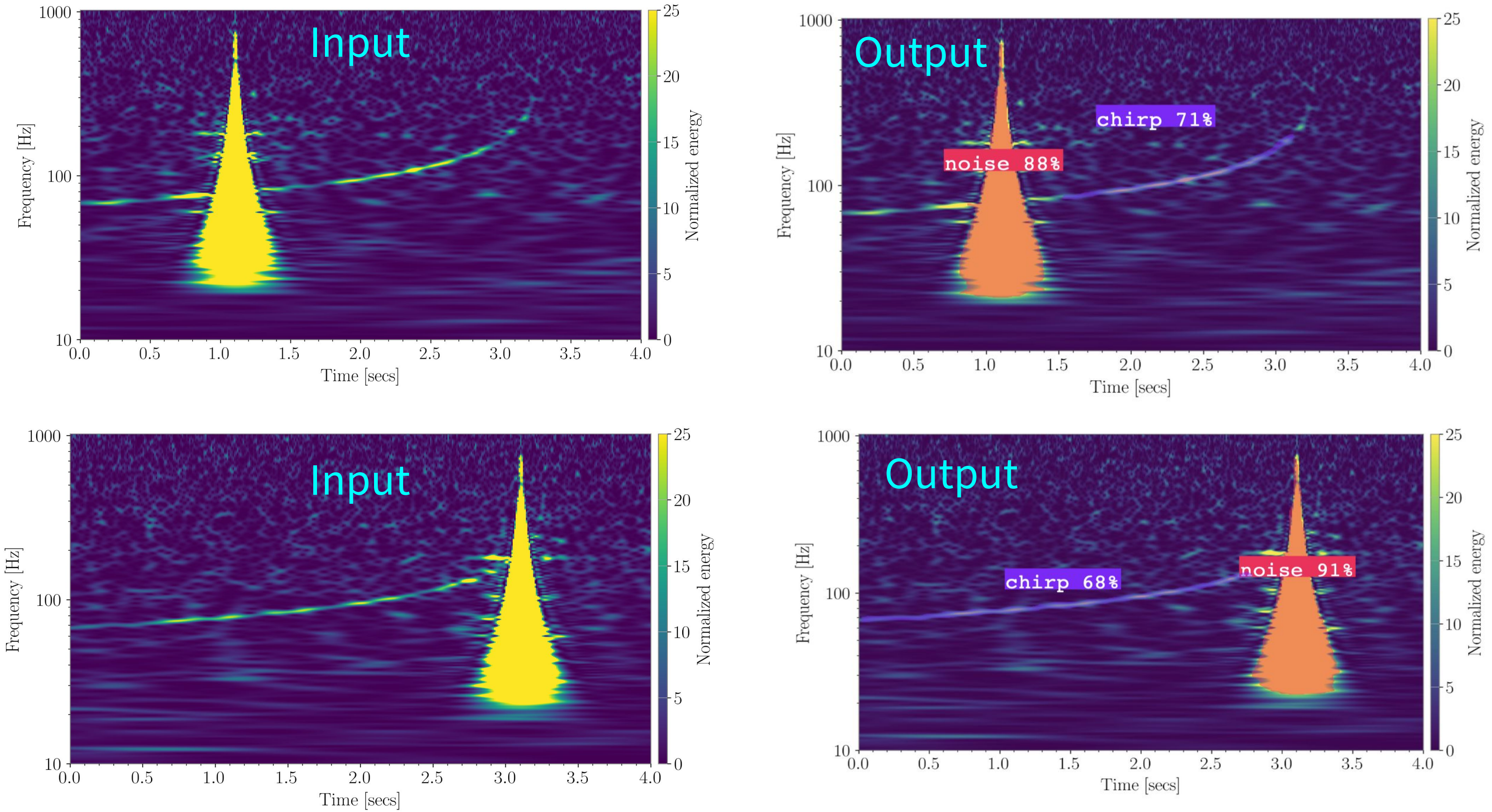
Image source

Example results (inference) : BBH chirps

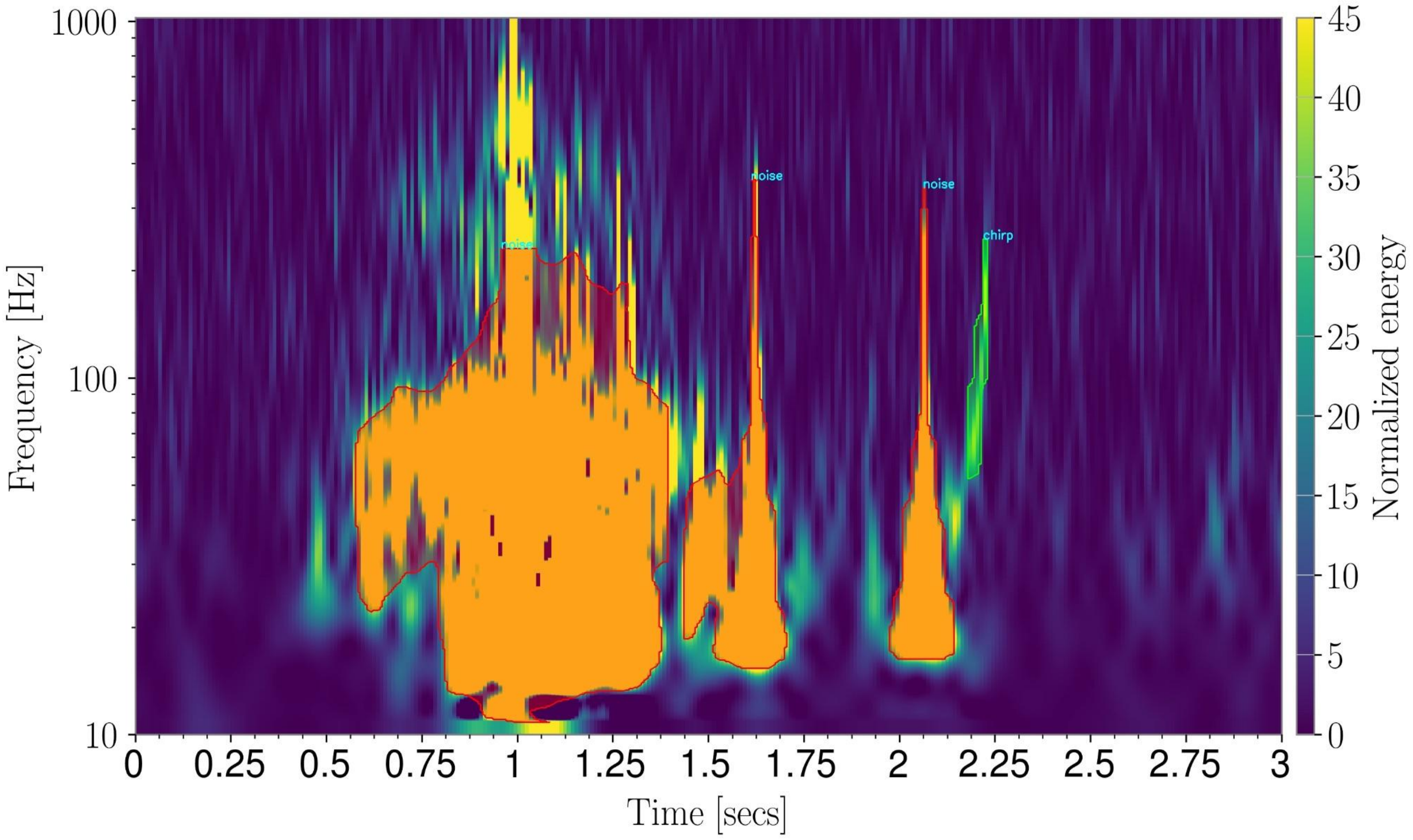


Source: https://zenodo.org/records/5649212, https://zenodo.org/records/7890437

Example results (inference): Chirps + Glitch

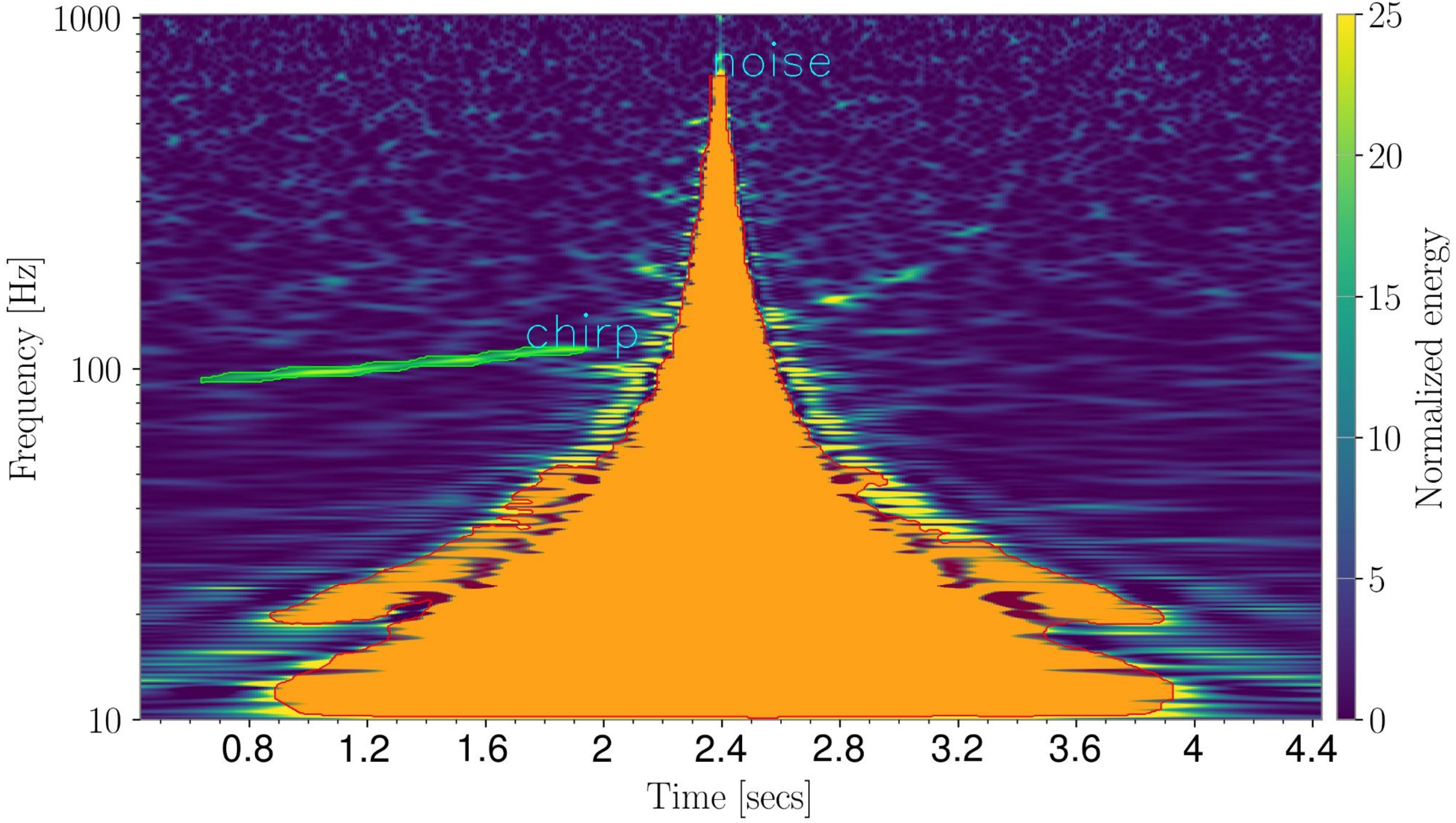


Source: https://zenodo.org/records/5649212, https://zenodo.org/records/7890437

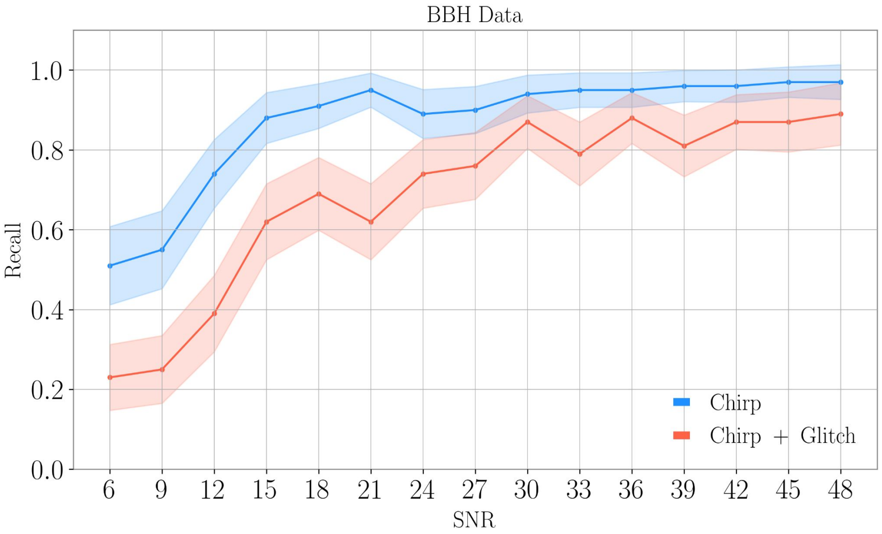


Source: https://zenodo.org/records/5649212, https://zenodo.org/records/7890437

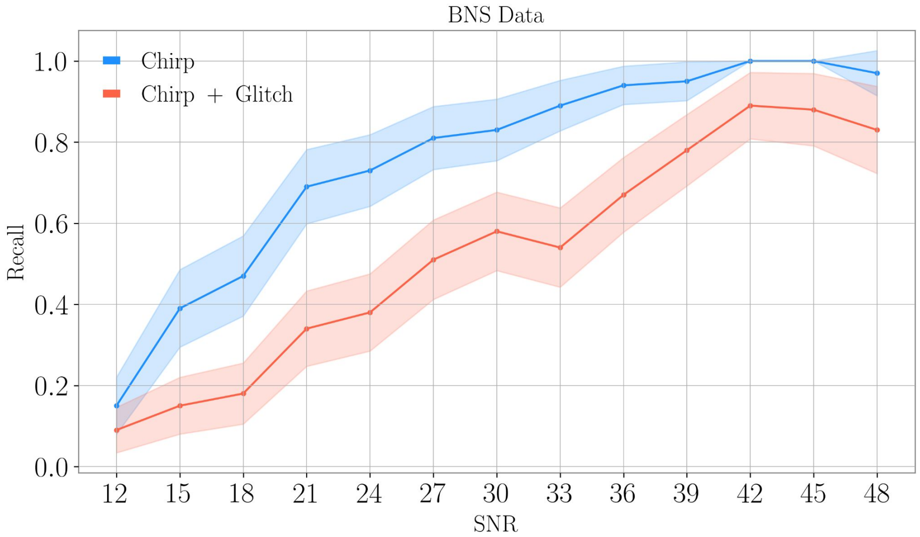
Example inference: GW170817



BBH Study



BNS Study



- Recall = TP/(TP + FN) : what fraction of data is correctly classified as Chirp
- Addition of glitches reduces recall
- Despite that, the model maintains a high level of performance
- Paper is on arXiv https://arxiv.org/abs/2508.17399

Summary

- Machine Leanring is extensively used in LIGO Detector Characterization and Data Analysis
- Improvements in ML Landscape are translating to LIGO applications
- Increased application of ML and automation is needed as our detectors get more sensitive and noisier
- Application of ML in LIGO hardware, design optimization, suspension design.

Thank You!
Questions?

Extra Slides

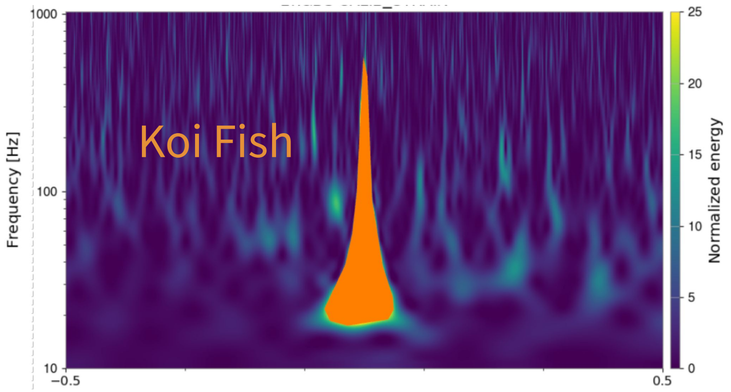
Apply YOLO to all the glitch categories

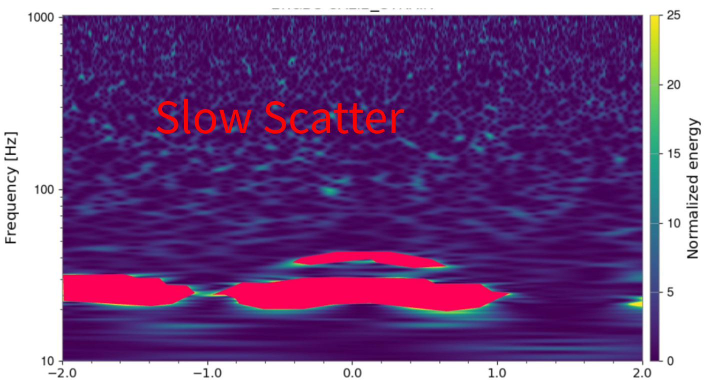
YOLO for the whole catalog of glitches

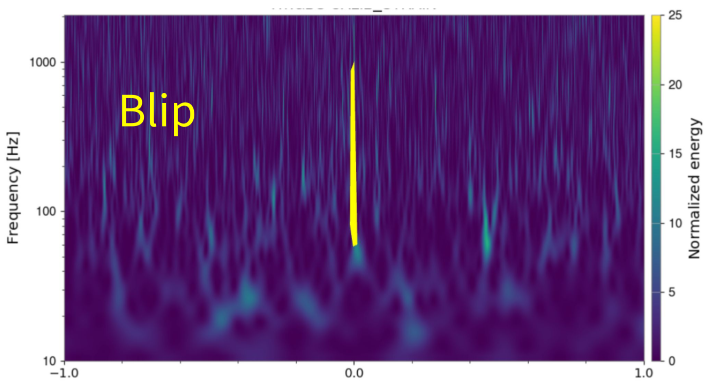
- Our current tool GW-YOLO is for distinguishing between chirps and transient noise
- Developing a new tool that will distinguish different categories of transient noise, like GravitySpy
- Motivation:
    - Enhanced characterization of transient noise due to segmentation capabilities (exact time frequency pixel masks)
    - Larger models are better at capturing richer features
    - Pretrained model which is fine tuned. Transfer learning leads to good performance even with limited training data

Preliminary Training Set

- Annotated images of glitches of different durations
- Some glitches are better resolved at low duration (Blips) while others need longer spectrograms (Slow Scatter)
- Current training dataset has Slow Scatter, Blips, Tomte, Koi Fish and Extremely Loud
- Training set is relatively small : about 250 images

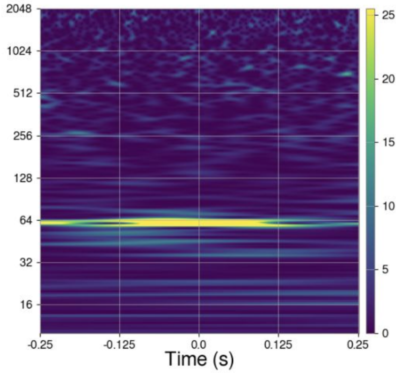


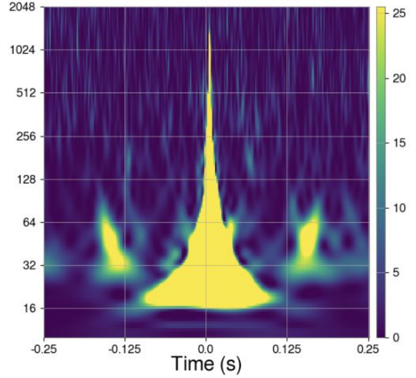


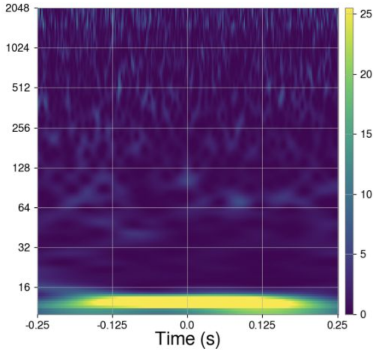


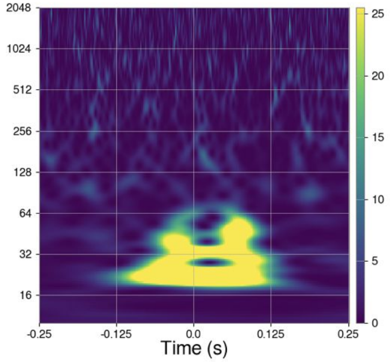
How many categories of transient noise?

- Before supervised training on YOLO, we need to know the number of different transient noise morphologies in the data
- Clustering, Autoencoders, tSNE can help with this
- Map high dimensional Q-transform data to low dimensions and count different clusters in low dim





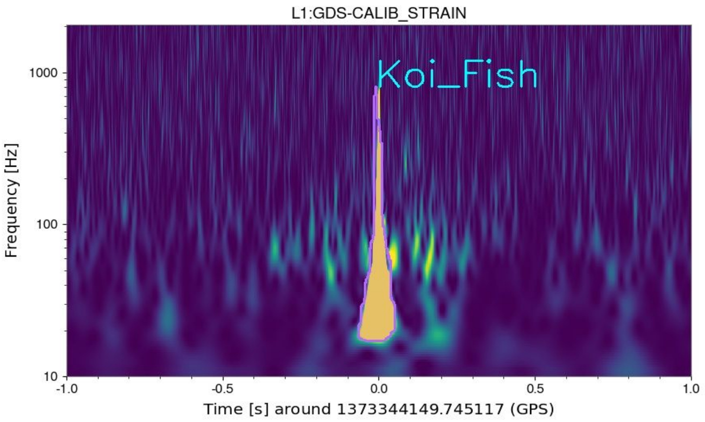


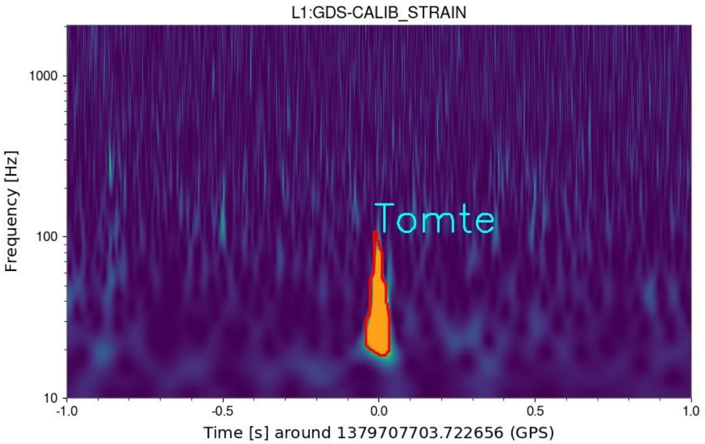


Inference on new data

- Trained the YOLOv8 Segmentation model on the data.
- Precision and Recall are about 82% on the validation set
- These numbers can be increased with more data.
- Inference on new data, performance looks pretty good.
- Made some changes to prepration of Q-transforms on new data.
- Will feed these changes back to training dataset preparation

More accurate time-frequency information





Tool	Glitch YOLO	GravitySpy/Omicron
Koi Fish Duration	0.12 sec	1 sec
Koi Fish Bandwidth	843 Hz	5237 Hz
Tomte Duration	0.08 sec	0.58 sec
Tomte Bandwitdh	64 Hz	7856 Hz

YOLO segmentation model provides much better time and frequency localization of the glitch morphology