Multi-transient segmentation in LIGO using computer vision

Siddharth Soni
Nikhil Mukund
Erik Katsavounidis

LIGO Lab MIT

Scientific Machine Learning
for Gravitational Wave Astronomy
June 2 - 6, 2025

Overview

- LIGO and Detector Characterization
- Computer Vision
- Computer Vision in LIGO
- Summary

Ground based Gravitational Wave detectors

Purpose: To detect GWs in the band 20 - 2000 Hz

Where: 2 LIGO detectors in USA, Virgo in Italy, KAGRA in Japan

Operations: Have completed 3 Observing runs (O1, O2, O3), O4 is currently ongoing

Detections: More than 200 detections have been made since the start of O1

LIGO Livingston, Louisiana, USA

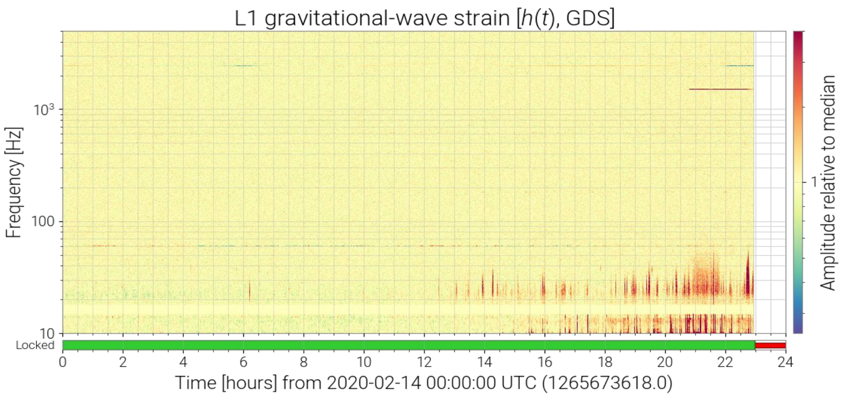
LIGO Hanford, Washington, USA

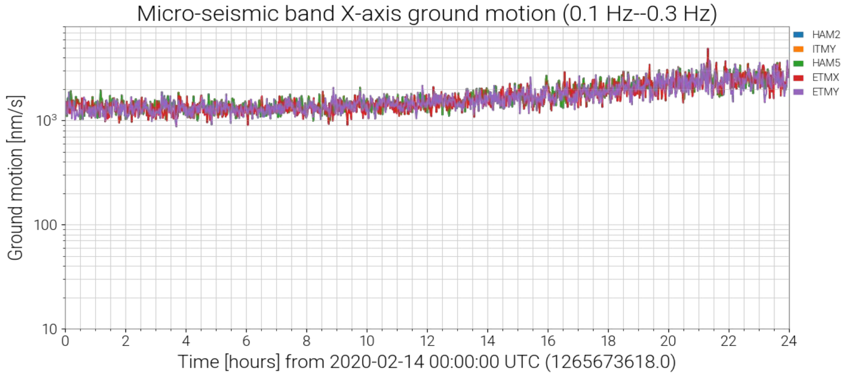
Virgo Cascina, Italy

https://www.ligo.caltech.edu/

Detector Characterization

- Monitor the status of detector : Instrumental and data quality investigations
- Physical Environment Monitoring: injection tests and noise coupling calculations
- Event Validation: Check data quality around the events
- Summary pages and Detchar tools: maintenance and development

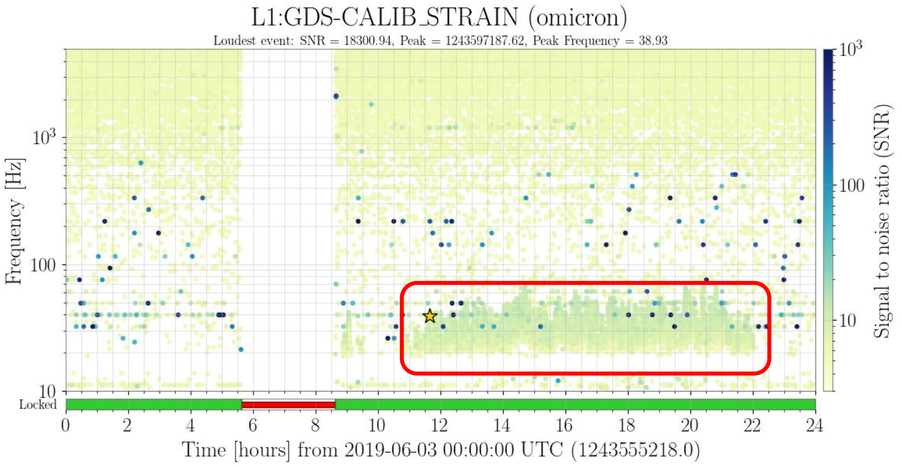


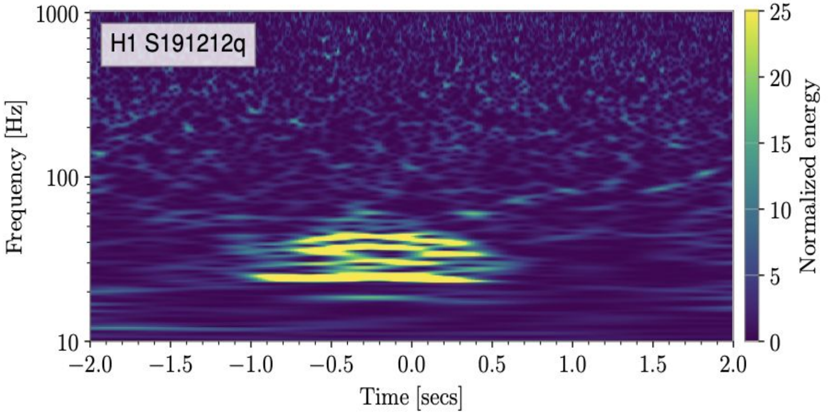


LIGO detector characterization in the first half of fourth Observing run. S Soni *et al* 2025 *Class. Quantum Grav.* **42** 085016

Transient noise aka glitches

- Short duration excess power
- Environmental or instrumental coupling
- Reduce sensitivity/range of the detector
- Mess up with the real events, parameter estimation, create false alerts, reduce sensitivity for stochastic searches
- Originate in detector hardware. Investigate using detchar tools, injections, on/off tests etc





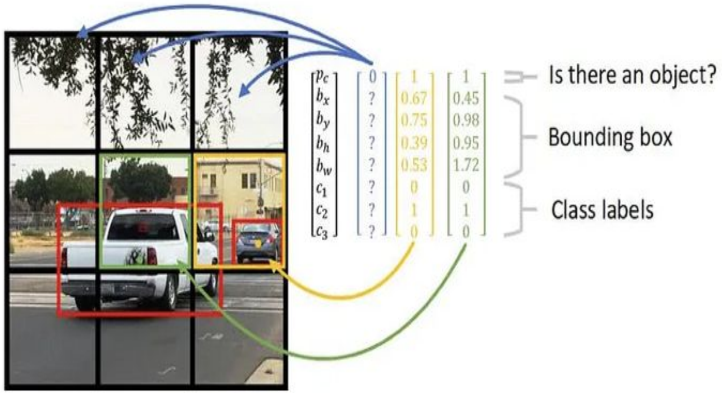
Computer Vision

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

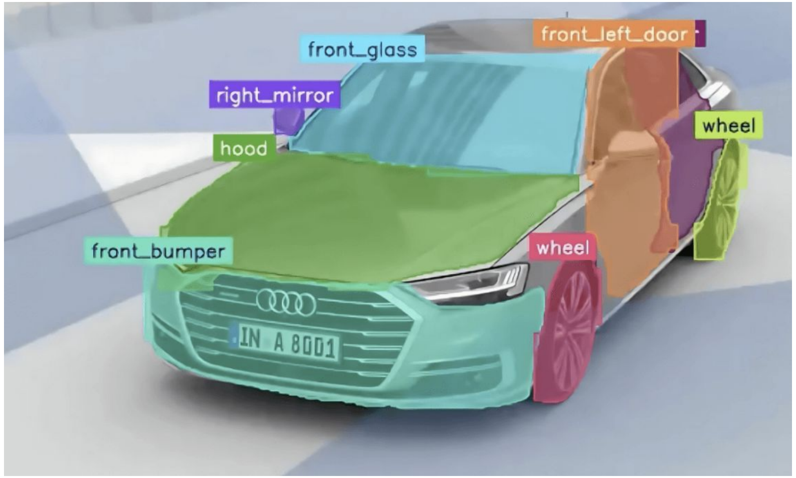


YOLO algorithm
https://arxiv.org/abs/1506.02640

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



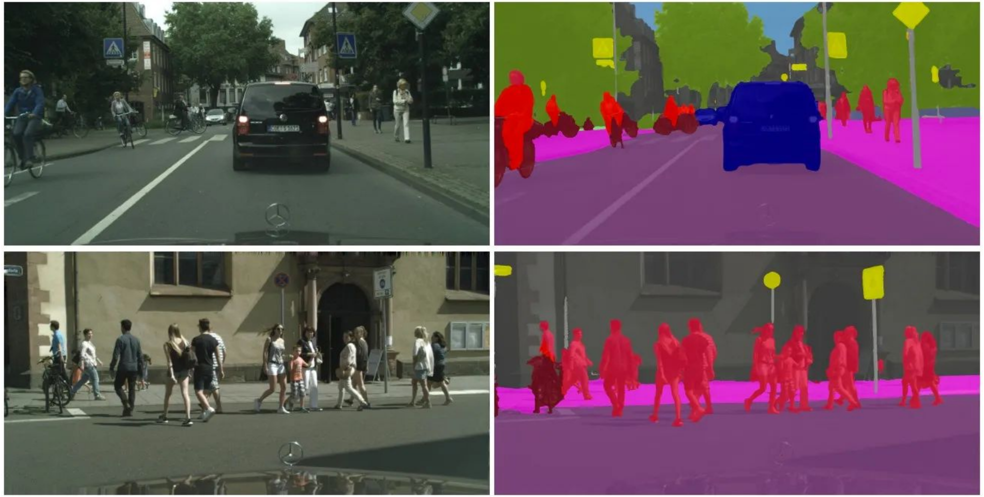
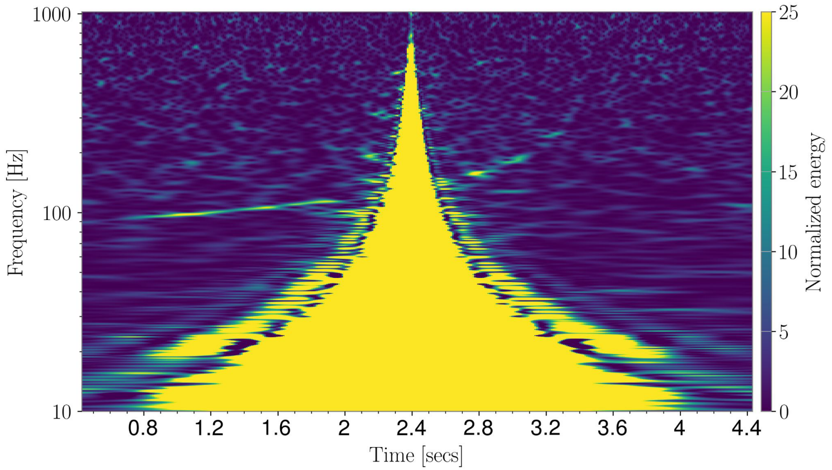


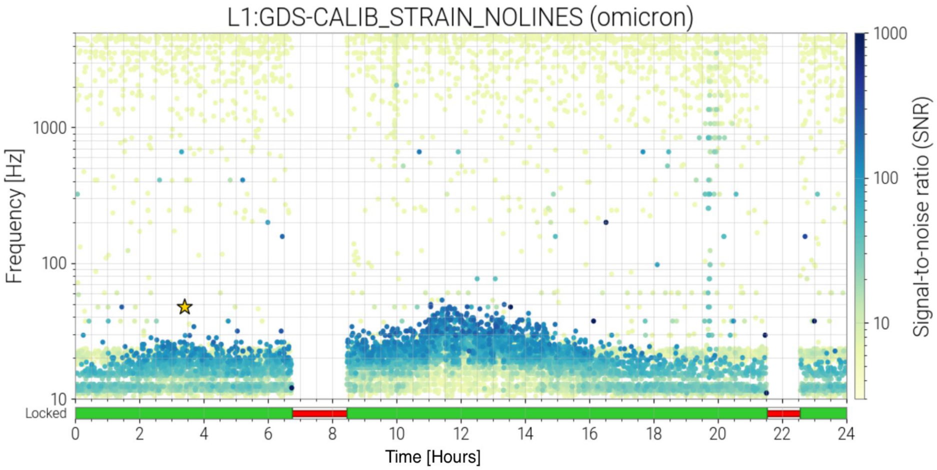
Image source, source

Computer Vision in LIGO

Motivation

GW170817





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

Main Idea

Spectrogram as Input → YOLO based Segmentation Model → Output Image: Class label (Noise or Chirp) & Segmentation masks

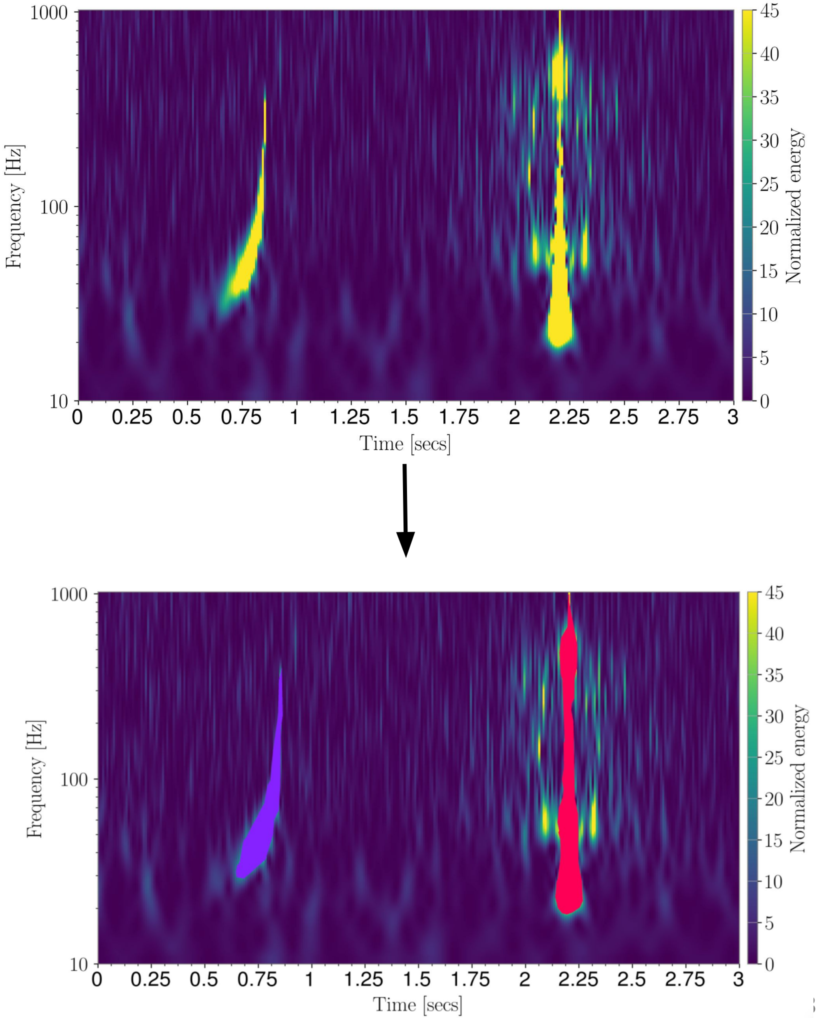
We feed Q transforms (or event times) to the script and it outputs a Q scan showing different class labels and segmented pixel-masks for each class

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 %

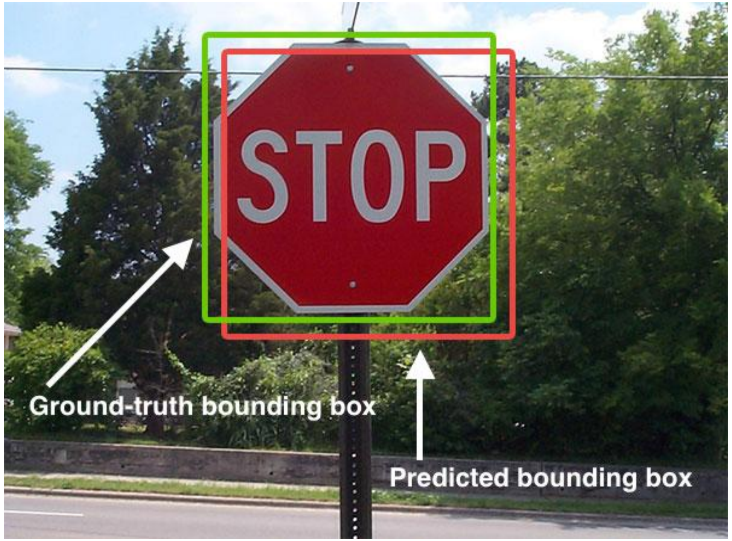
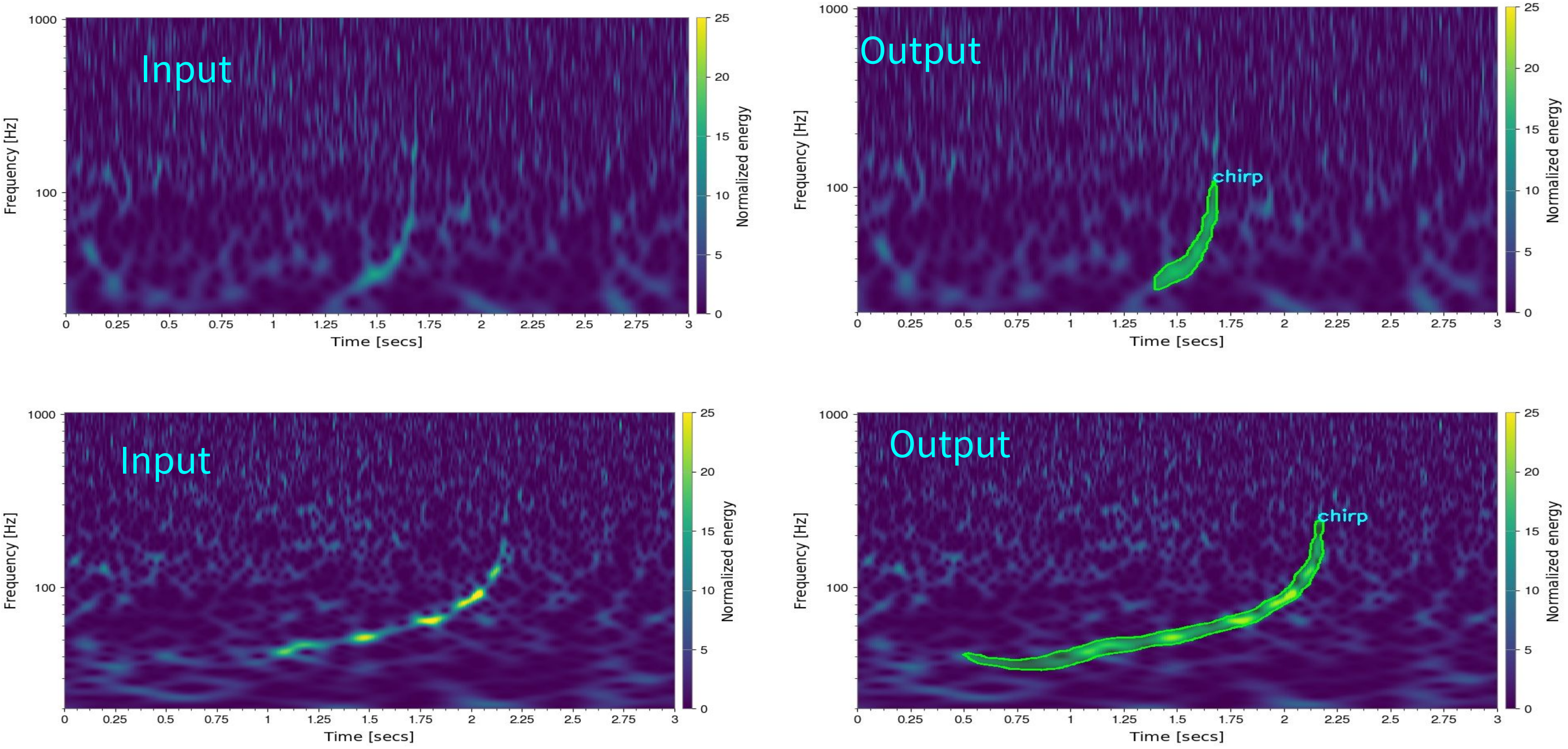


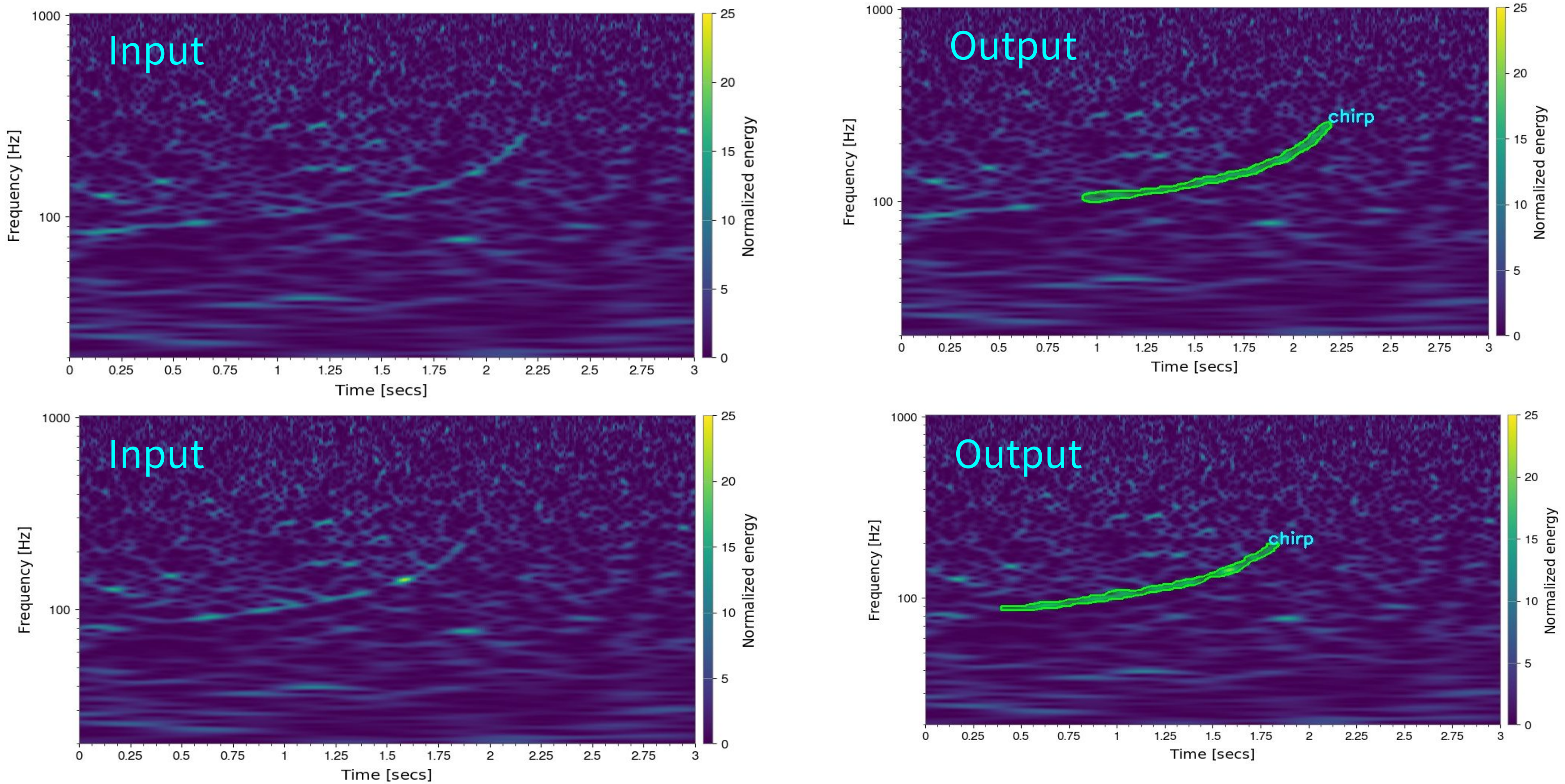
Image source

Example results (inference) : BBH chirps



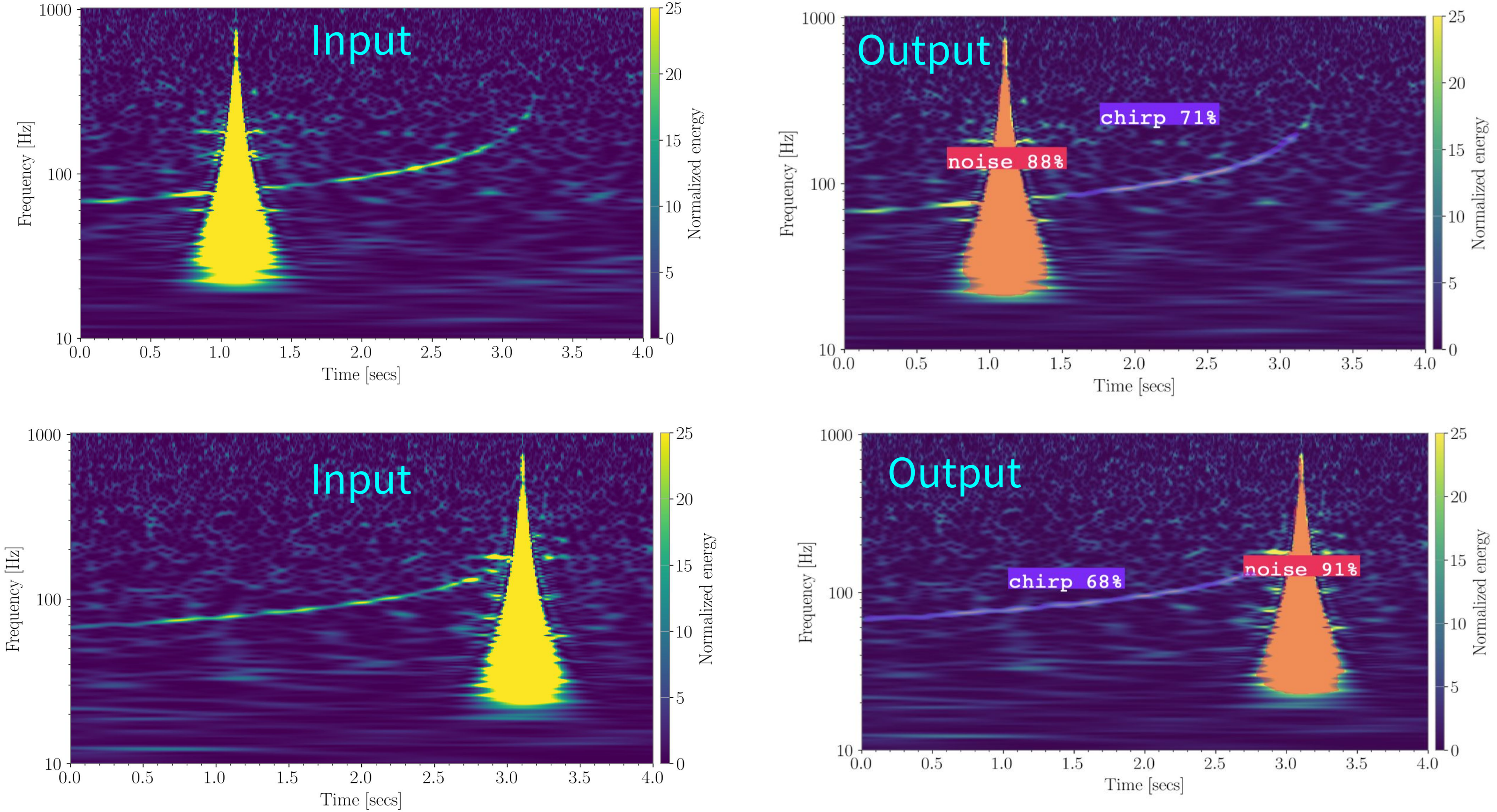
Source: https://zenodo.org/records/5649212, https://zenodo.org/records/7890437

Example results (inference): BNS chirps

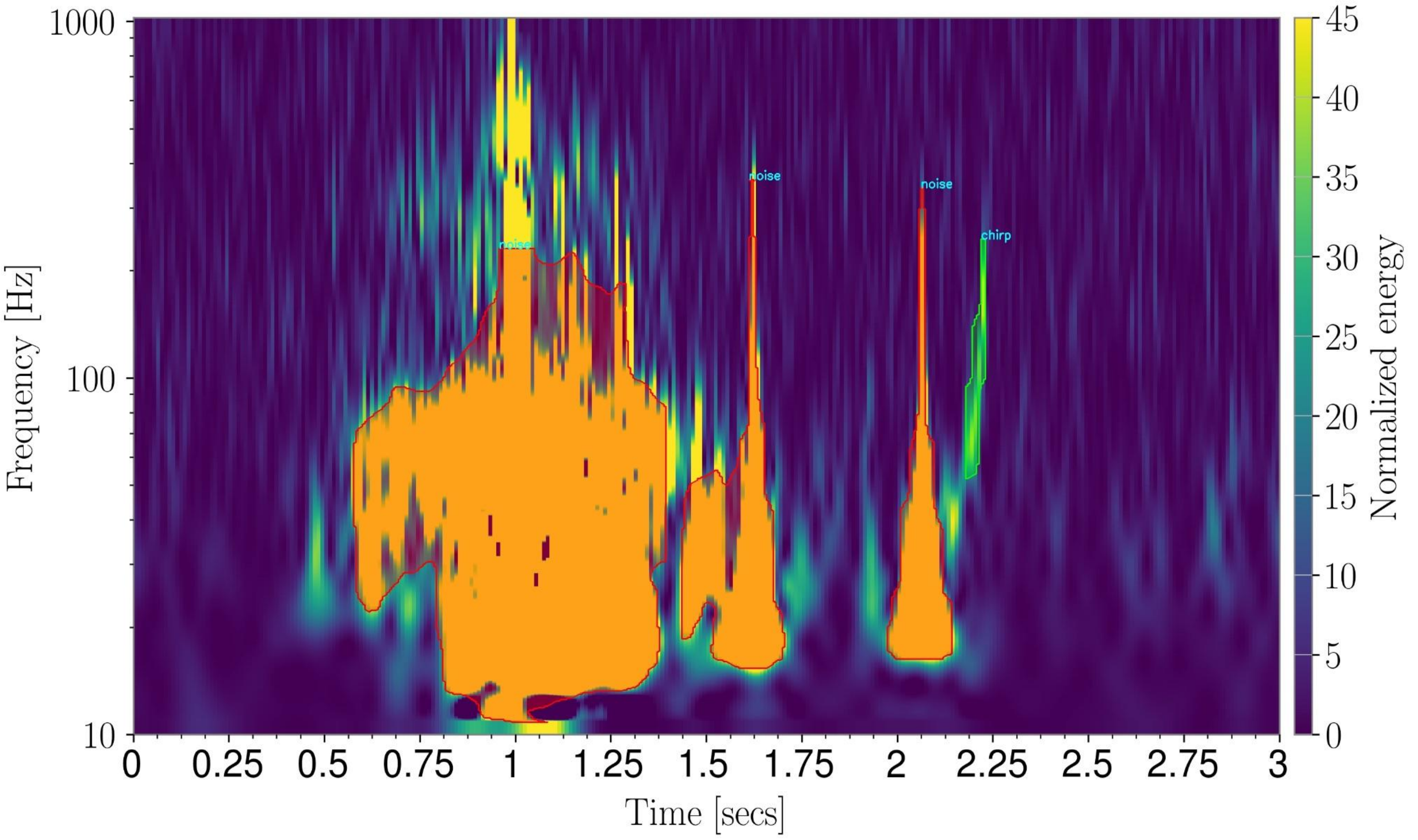


Source: https://zenodo.org/records/5649212, https://zenodo.org/records/7890437

Example results (inference): Chirps + Glitch



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

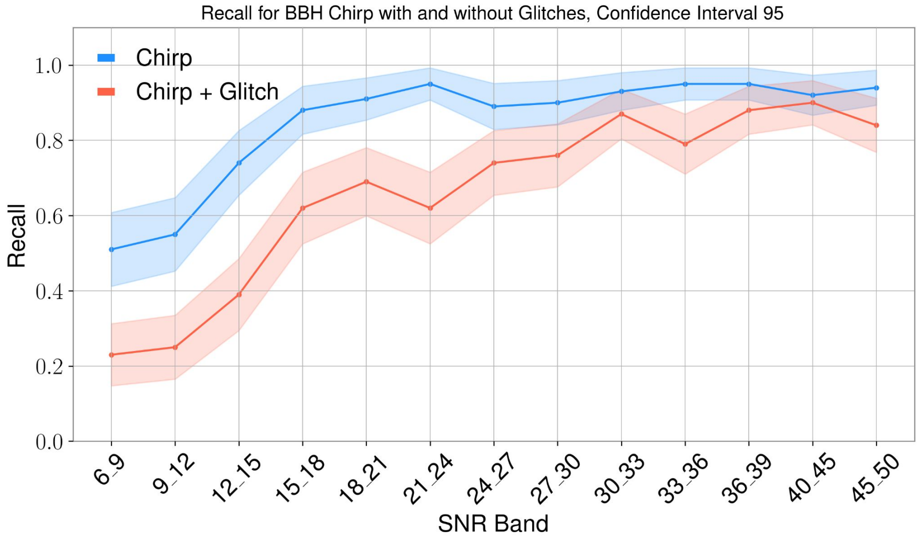


Source: https://zenodo.org/records/5649212, https://zenodo.org/records/7890437

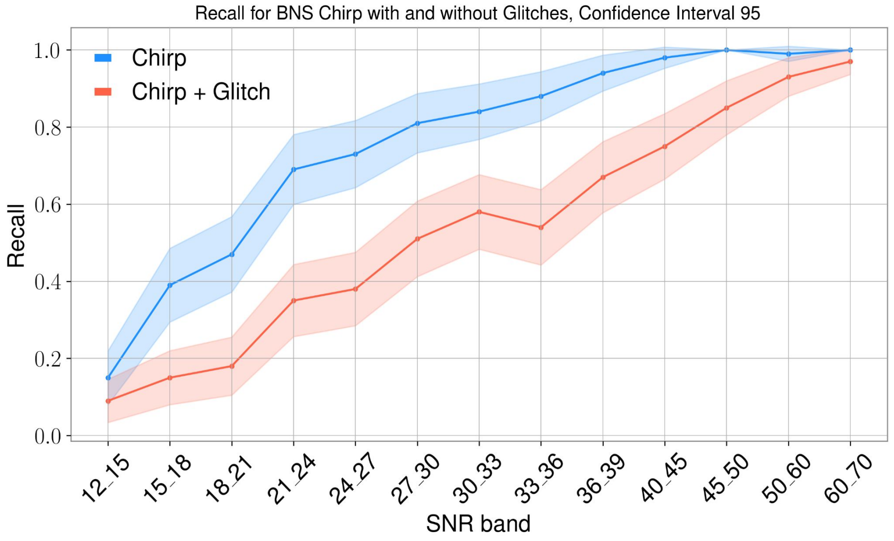
Large Sample Inference study

- Single-detector analysis (used LIGO-Livingston data only)
- BBH chirps from SNR 6 to 50 divided into multiple SNR bands
- BNS Chirps from SNR 12 to 70 divided into multiple SNR bands
- Glitch population is randomly sampled with SNR above 7.5 from O3 data
- Four datasets: BBH Chirps, BBH Chirps + Glitch, BNS Chirps, BNS Chirps + Glitch
- Measuring recall- what fraction of data is correctly classified as Chirp
- Around 1300 examples for better statistical results

BBH Study



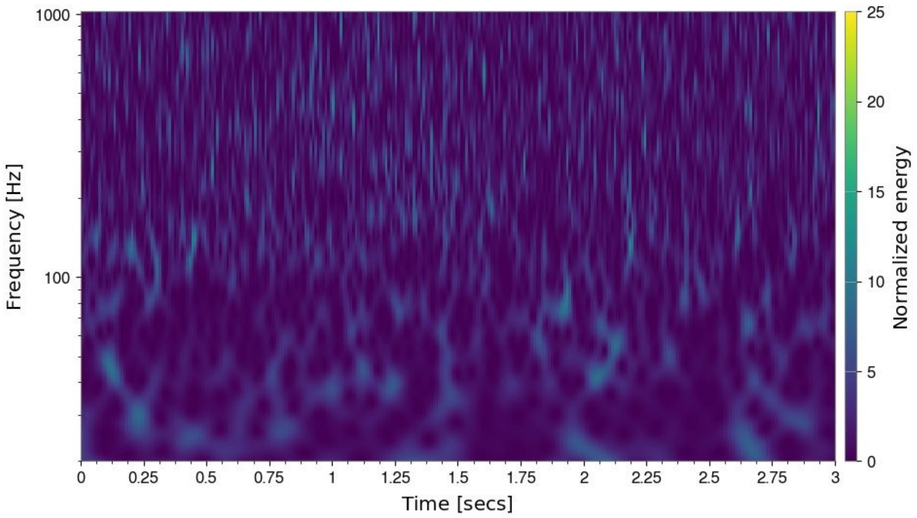
BNS Study

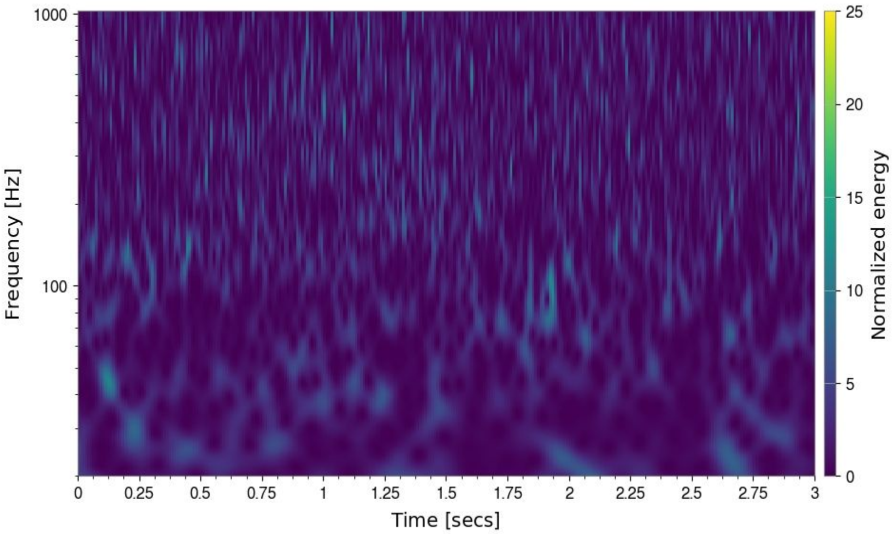


- Recall = TP/(TP + FN) : what fraction of data is correctly classified as Chirp

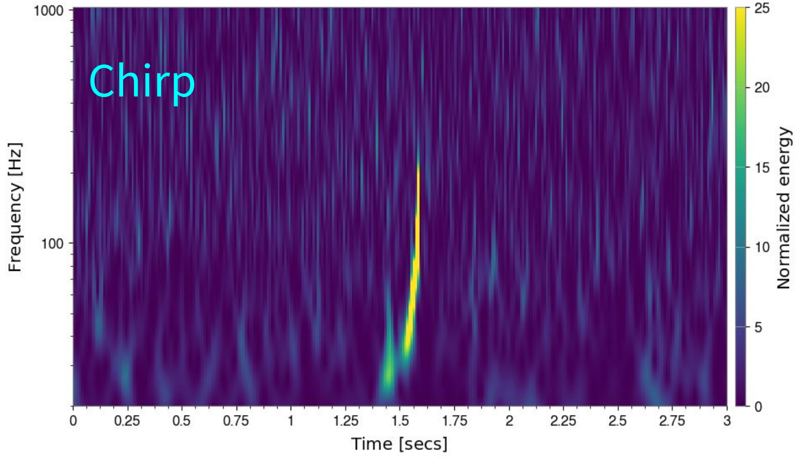
- Addition of glitches reduces recall
- Despite that, the model maintains a high level of performance

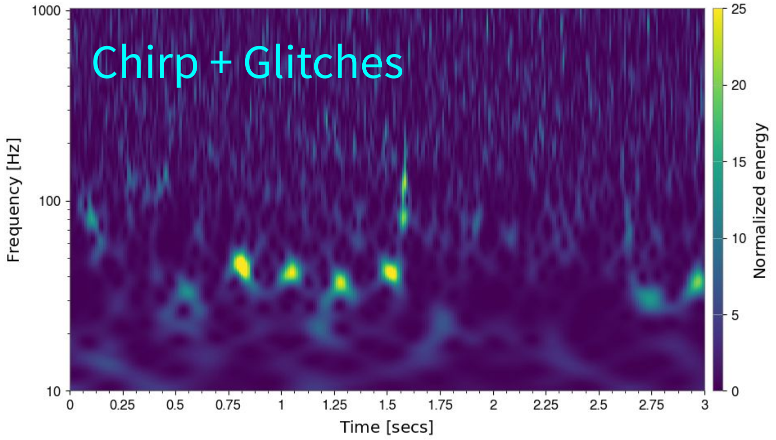
No chirp detected

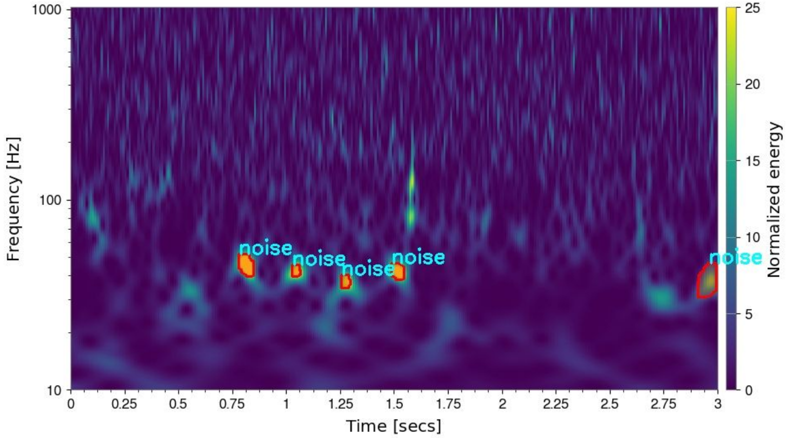




No chirp detected

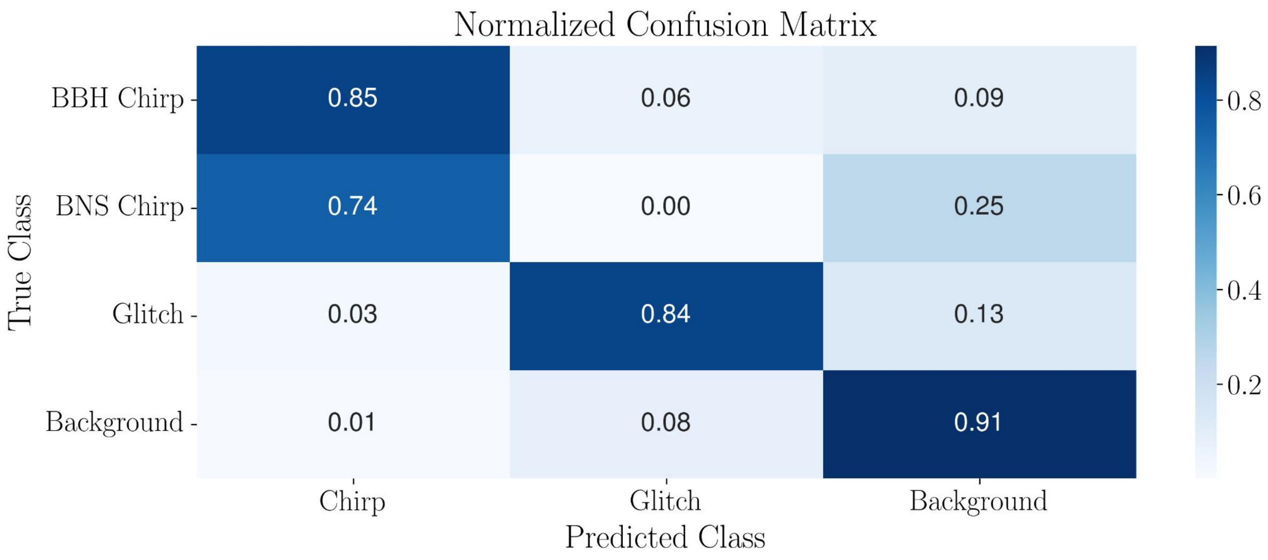




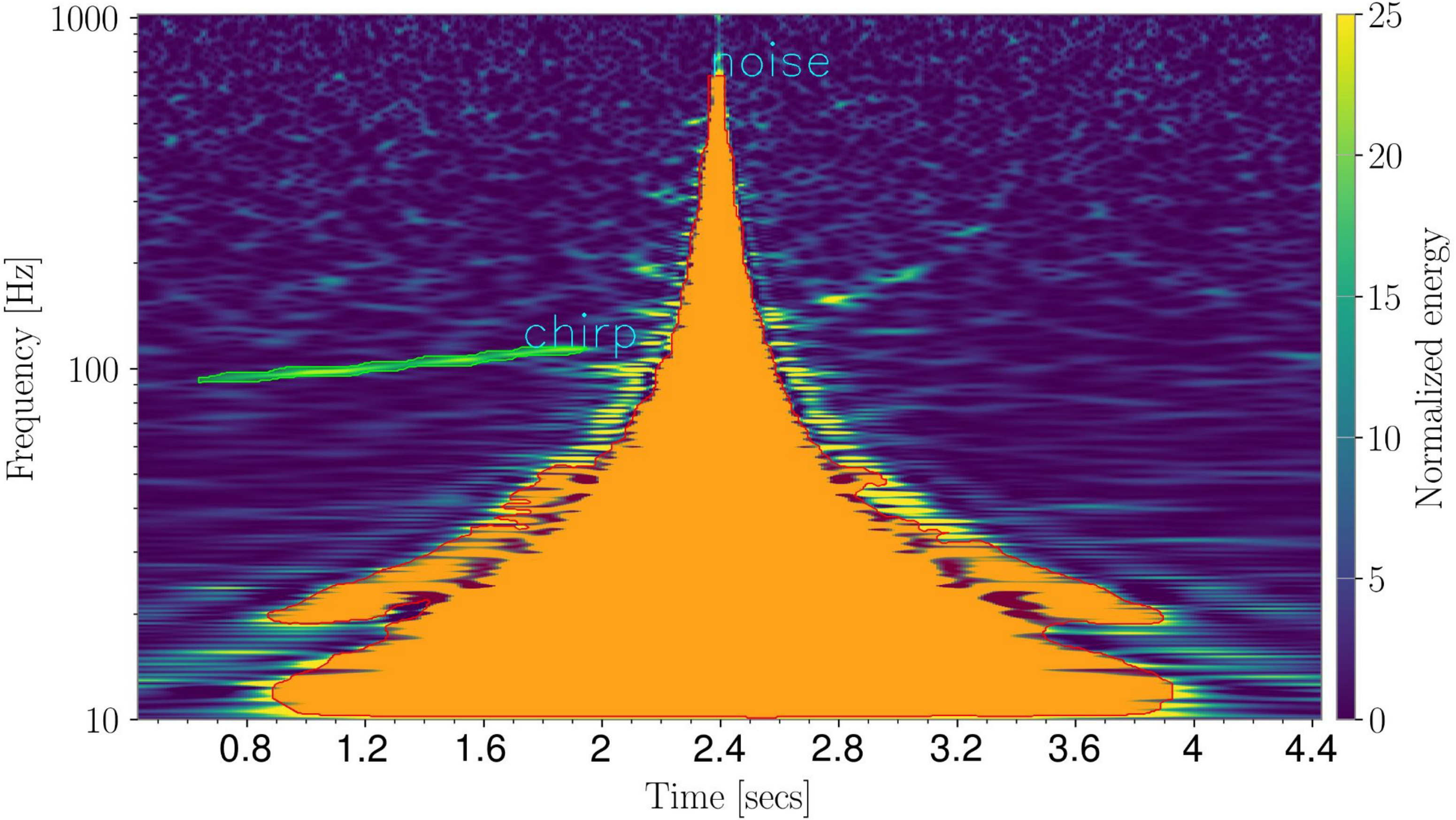


Confusion Matrix

- Previous plots show the ability of the model to identify chirps, given there are chirps in the data
- We also want to understand the general performance
- Background is defined as instrument data devoid of transient events (no signals or glitches)



Example inference: GW170817



Summary and Outlook

1. Image segmentation model can successfully **identify** and **localize** transient events in gravitational-wave Q-scan spectrograms
2. Method can **delineate transient events down to the pixel level**, providing analytic information for each one of them, including time-frequency localization/specification
3. **Multi-object identification** particularly powerful in gravitational-wave transient event analysis in the case of:
    a. signal + noise, e.g., binary coalescences in proximity/overlap with glitches
    b. multiple signals occurring in close proximity, e.g., high rate of binary events resulting to almost overlapping events, lensing events
4. Algorithm can **run at near zero-latency**, once Q-transform and YOLO pass data completely in memory (currently intermediate file I/O is used)
5. Method effectively **replaces human(eye)-in-the-loop for at-scale applications**; offers quantitative statements, accompanied by efficiency and false alarm rates associated with them that can assist in:
    a. noise mitigation by offering improved localization of transient noise near the signal thus leading to automated noise subtraction
    b. overall efforts for automating Gravitational-Wave event validation

References

1. YOLO: https://arxiv.org/abs/1506.02640
2. Ultralytics: https://www.ultralytics.com/
3. GW170817: https://arxiv.org/abs/1710.05832
4. Q-transform: https://dspace.mit.edu/handle/1721.1/34388
5. LIGO Detector Characterization in the first half of fourth Observing run: https://arxiv.org/abs/2409.02831
6. O3 Injections Dataset: https://zenodo.org/records/7890437
7. PyCBC: https://pycbc.org
8. Noise dataset: https://zenodo.org/records/5649212

Acknowledgement:

- This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation.
- LIGO Lab - PHY-2309200
- A3D3-PHY-2117997

Thank You!
Questions?

Extra Slides

YOLO Model Architecture

BackBone

Extract features from input: edges textures, shapes

Summarizes “what” is “where”

Neck

Acts as a bridge between Backbone and Head

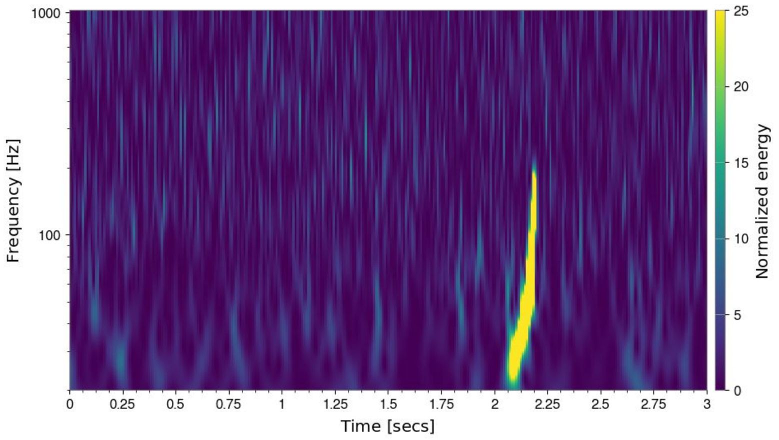
Provides contextual information

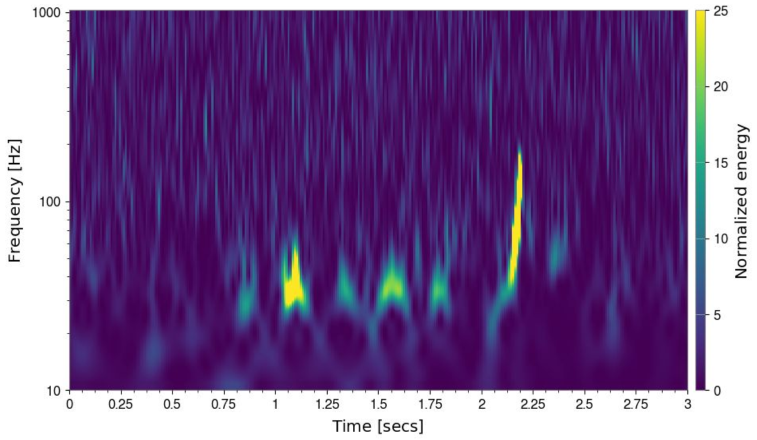
Merges information of different size features

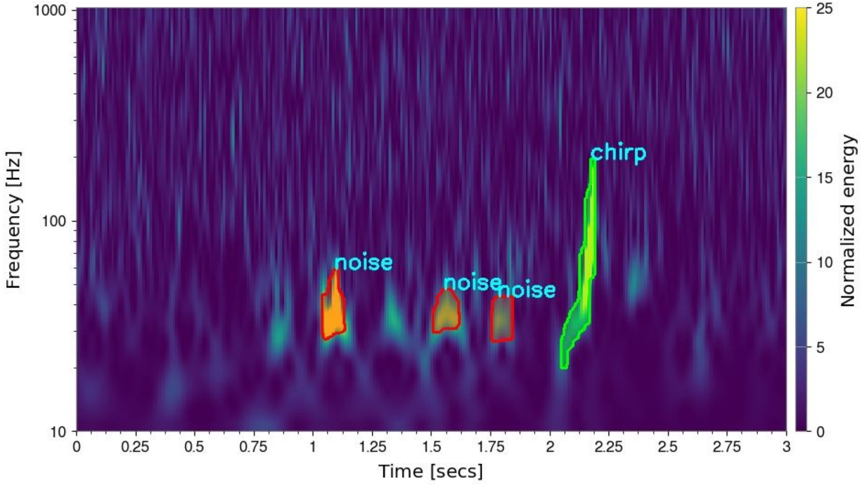
Head

Generates network outputs: bounding boxes, pixel masks, class labels

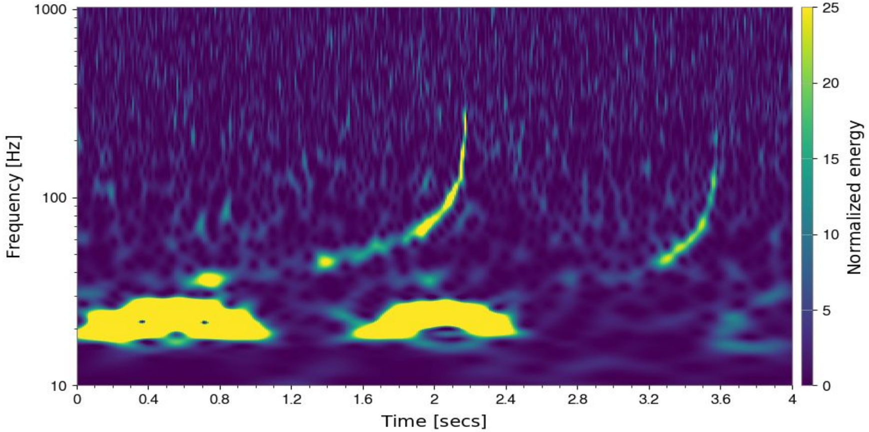
Transforms the feature maps prepared by Neck and Backbone into detections

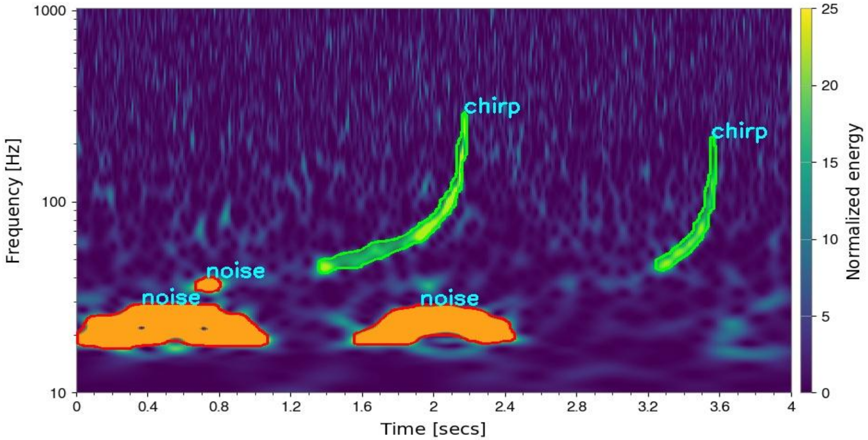

Frequency [Hz]
Time [secs]
Normalized energy

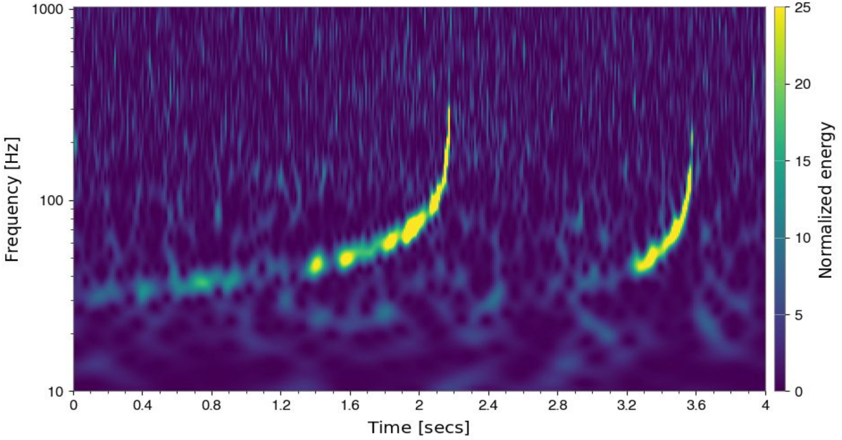

Frequency [Hz]
Time [secs]
Normalized energy

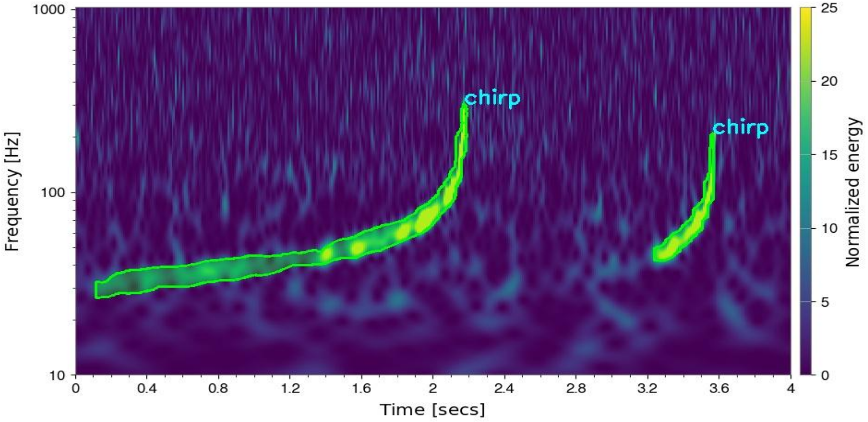

noise
noise
noise
chirp
Frequency [Hz]
Time [secs]
Normalized energy

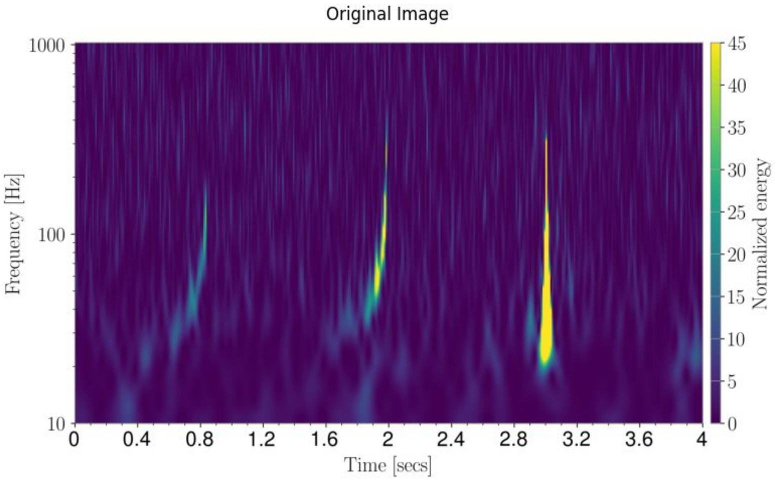
Multiple Chirps + Noise

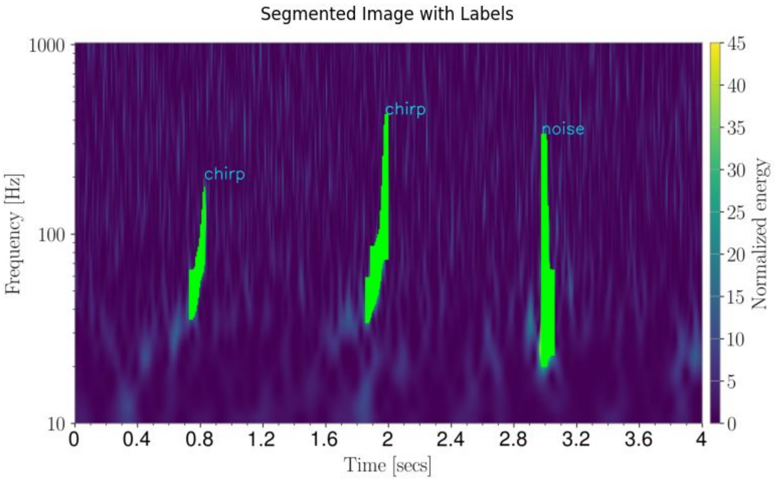









Original Image
1000
100
10
Frequency [Hz]
0
0.4
0.8
1.2
1.6
2
2.4
2.8
3.2
3.6
4
Time [secs]
45
40
35
30
25
20
15
10
5
0
Normalized energy


Segmented Image with Labels
chirp
chirp
noise
1000
100
10
Frequency [Hz]
0
0.4
0.8
1.2
1.6
2
2.4
2.8
3.2
3.6
4
Time [secs]
45
40
35
30
25
20
15
10
5
0
Normalized energy