This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 952480

PyMerger: Detecting Binary Black Hole mergers from the Einstein Telescope Using Deep Learning

Wathela Alhassan[1]

Advisor: Prof Tomasz Bulik[1,2] , Dr. M. Suchenek[1]

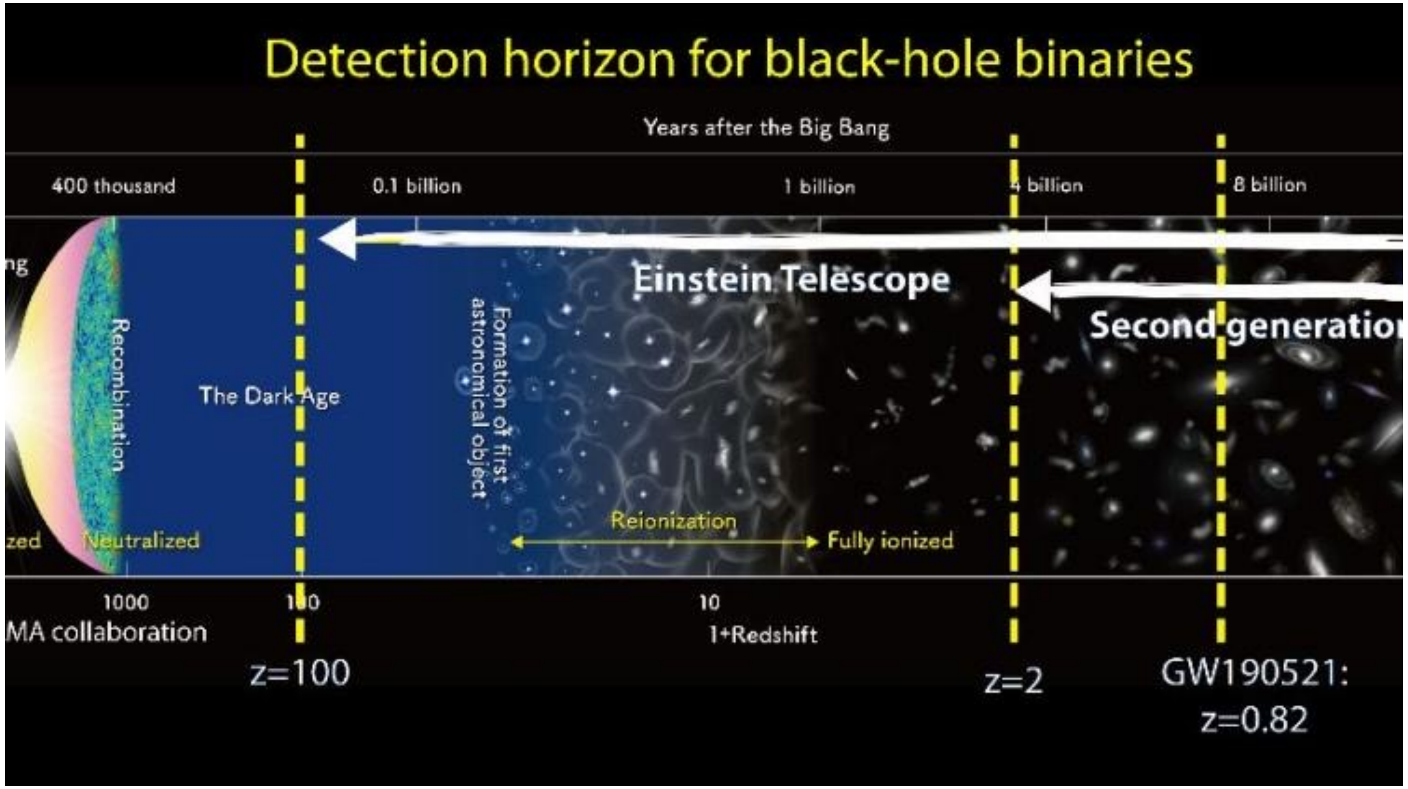
[1]Particle Astrophysics Science and Technology Centre, Nicolaus Copernicus Astronomical Center,

[2]Astronomical Observatory of the University of Warsaw

Statistics Journal Club, December 4th 2024, Astronomical Observatory, University of Warsaw.

3^{rd} generation detector – Einstein Telescope (ET)

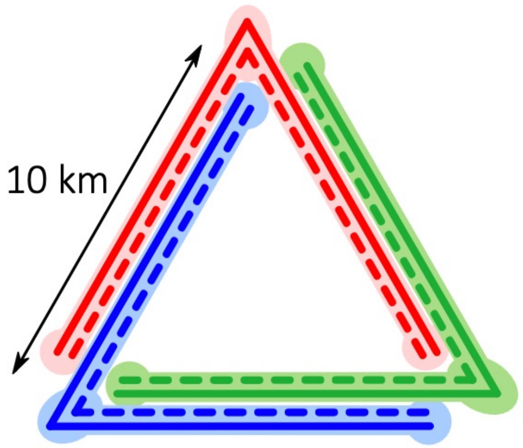
- Better sensitivity of one order of magnitude allowing detection at lower frequency.
- Annual detection rate for BBHs and BNSs of order $10^5 - 10^6$ and 7×10^4 respectively.
- Detection horizon for BBHs up to redshift z = 100
- BBHs with total solar mass of 20 - 100 will be visible up to ≈ 20



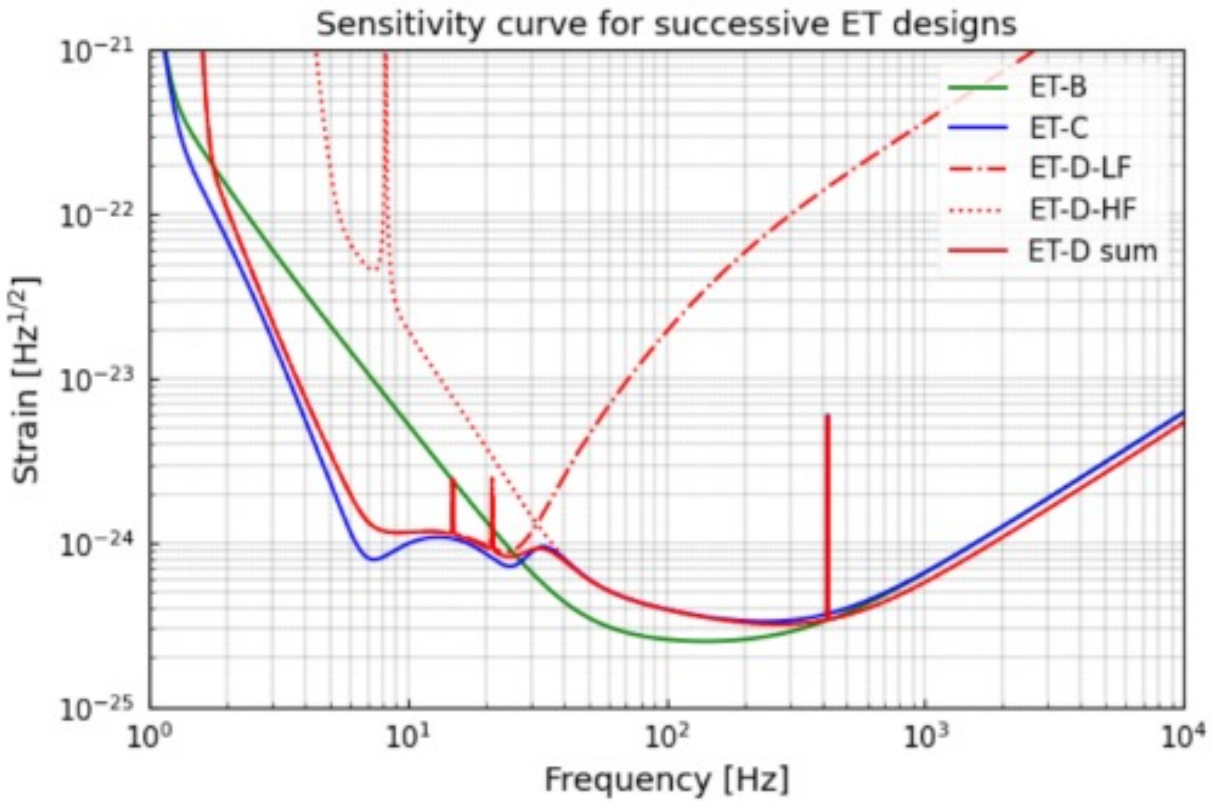
Credit: Sciencespring.com

3rd generation detector – Einstein Telescope (ET)

- ET-D design configuration
- Consist of three nested detectors (shown in blue, green and red) in a triangular arrangement.
- Each detector consists of two interferometers, one optimised for low-frequency (solid) and one for high-frequency sensitivity (dashed).



Credit: et-gw.eu



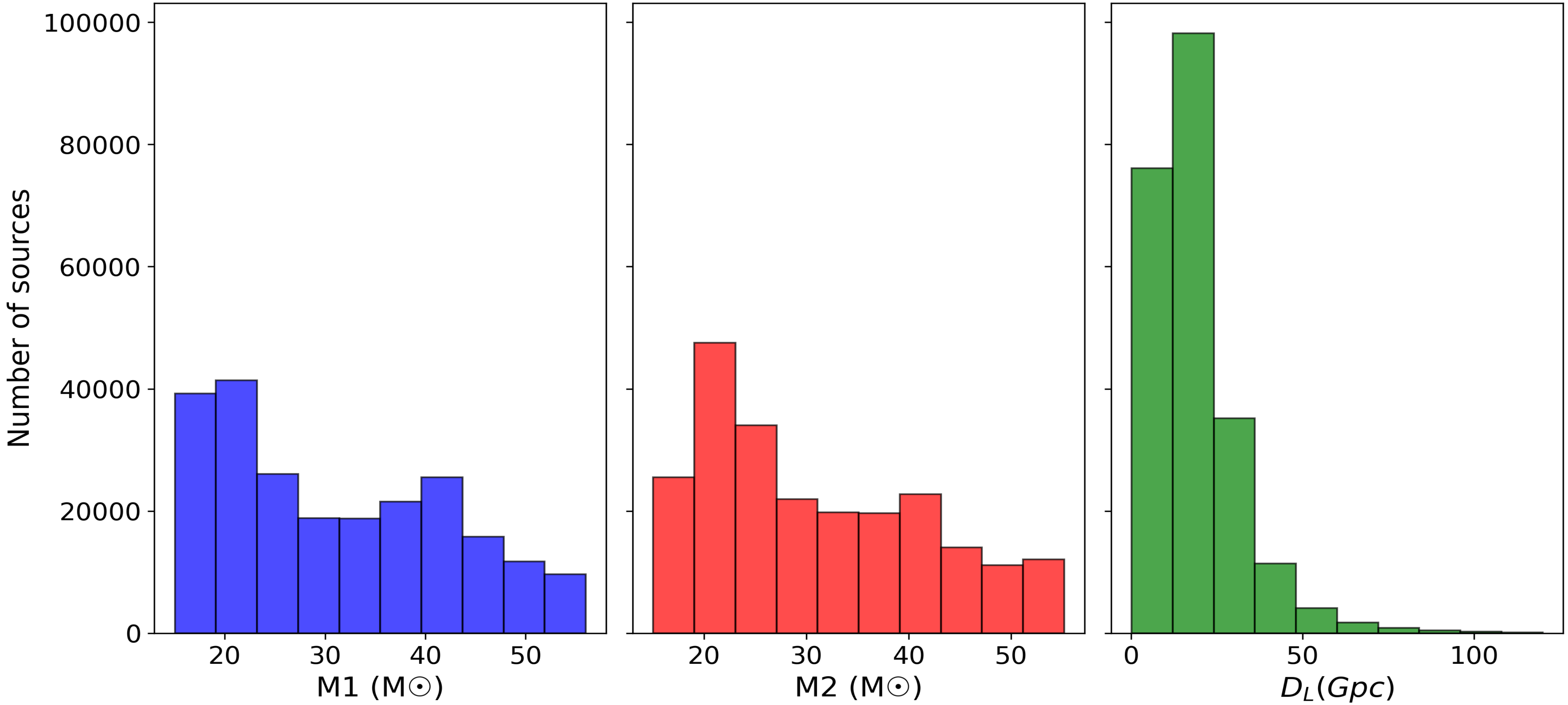
Motivation

- **What?** Investigate the efficiency of utilizing the time-frequency domain for detecting BBHs in ET data using Convolutional Neural Networks (CNNs).

- **Why?**
  - High volume of data is expected from ET.
  - Traditional GWs search methods such as match filtering will become impractical. This is due to the large template bank required and the difficulties in waveforms modelling.
  - The generalization ability of Deep Learning presents a promising alternative for gravitational wave data analysis, encompassing both detection and parameter estimation.

Simulation

- Parameters from Belczynski et al. 2020 generated using the population synthesis code StarTrack (Belczynski et al. 2002b,c,a).
- Focus on low and medium BBHs.



Simulation

Experiments:

- **Single Subdetector Data (SSDD):** Data simulated using only a single subdetector of the ET.
- **Three Subdetector Combined Data (TSDCD):** Data simulated using all three subdetectors of the ET combined.

Parameters	Values
Detector	SSDD: E1 | TSDCD: E1 + E2 + E3
RA and Dec	Random (uniform distribution)
Sources type	BBH
M1 and M2	$\in$ 15-56 $M_{\odot}$
Distance	140 – 120,000 Mpc
Inclination angel	Random choice between 0.5 and pi
Starting frequency (f_{low})	SSDD: 30 Hz | TSDCD: 5, 10, 15, 20 and 30 Hz
Time step (t_{Δ})	1.0/16384
Polarization phase	Random choices between 0.5 and 2 pi
Coalesence phase	Random choices between 0.5 and 2 pi

Short Time Fourier Transform (STFT)

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-i\omega t} d(t)$$

x(t): signal.
w: window function.
τ, ω: time and frequency axis.

- Window: $blackman$ window.
- 1024 length and 50% overlap.
- 62.5 ms temporal resolution.

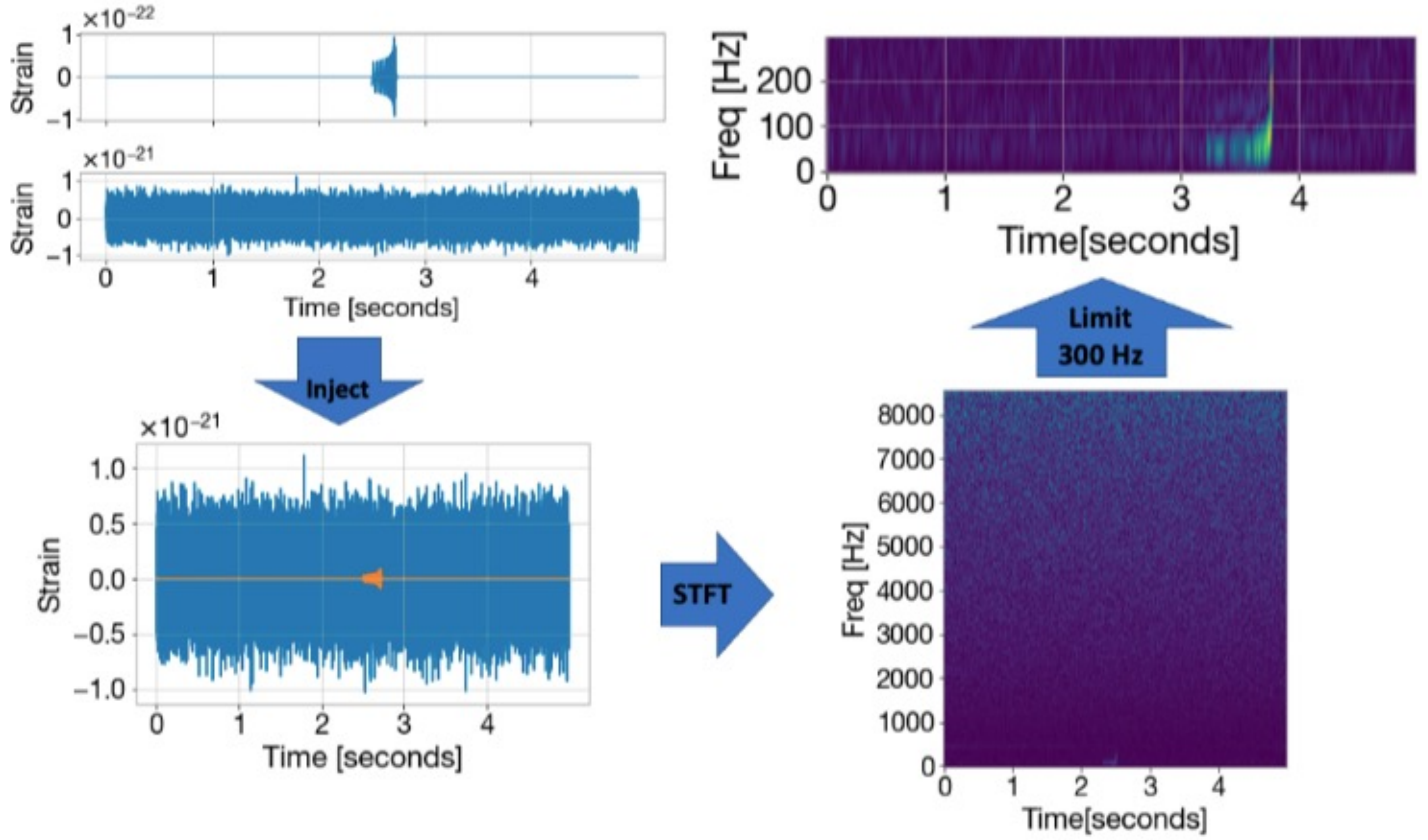
SSDD Simulation

$$d(t) = \begin{cases} h(t) + n(t), & \text{if data contains a BBH's signal} \\ n(t), & \text{if data contains noise only.} \end{cases}$$

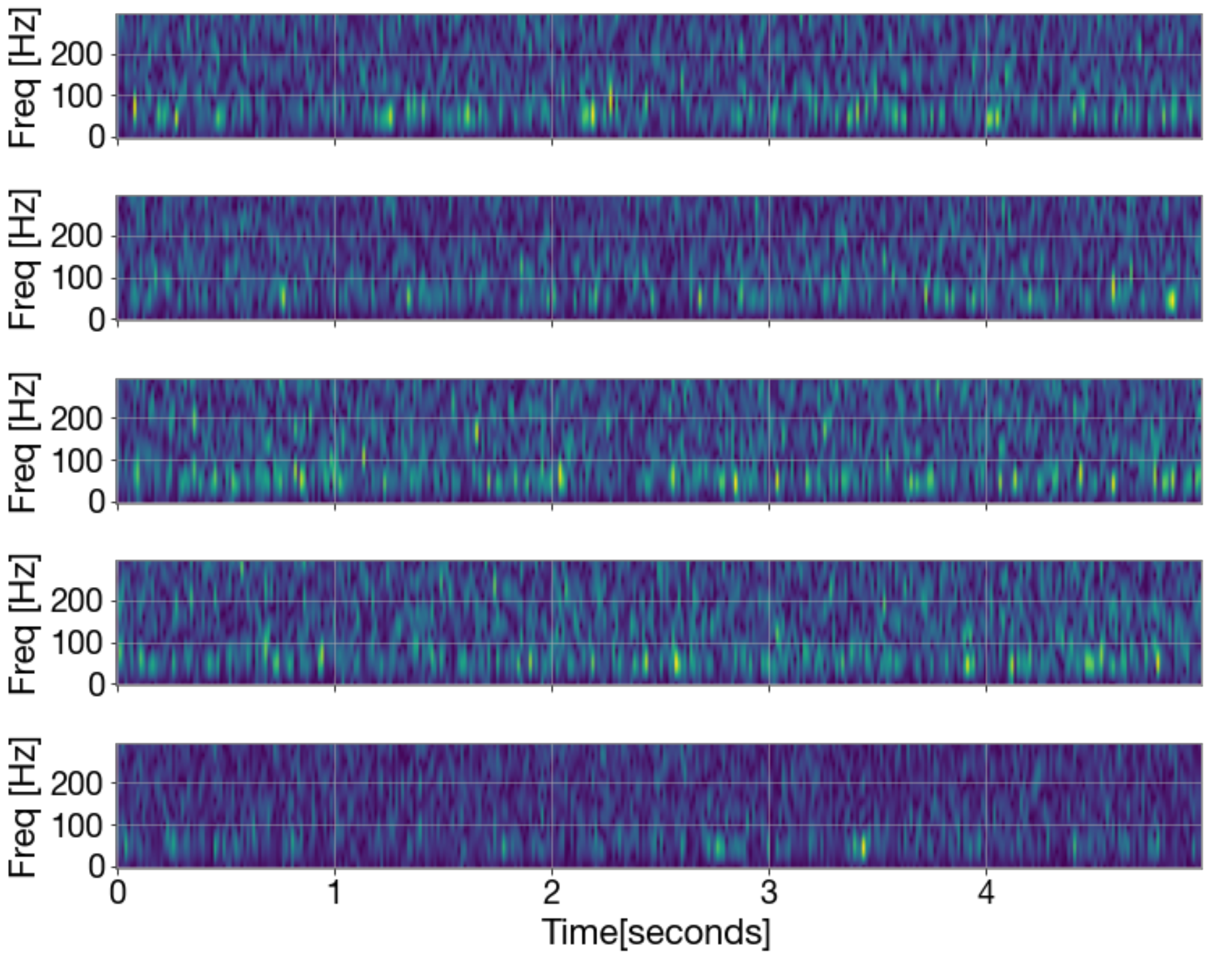
d(t): single-channel time-series data
h: BBH signal
n: gausian noise

- 25,000 BBHs injections.
- f_{low} : 30 Hz.
- SNR ranges: 4-5, 5-6, 6-7, 7-8 and >8.

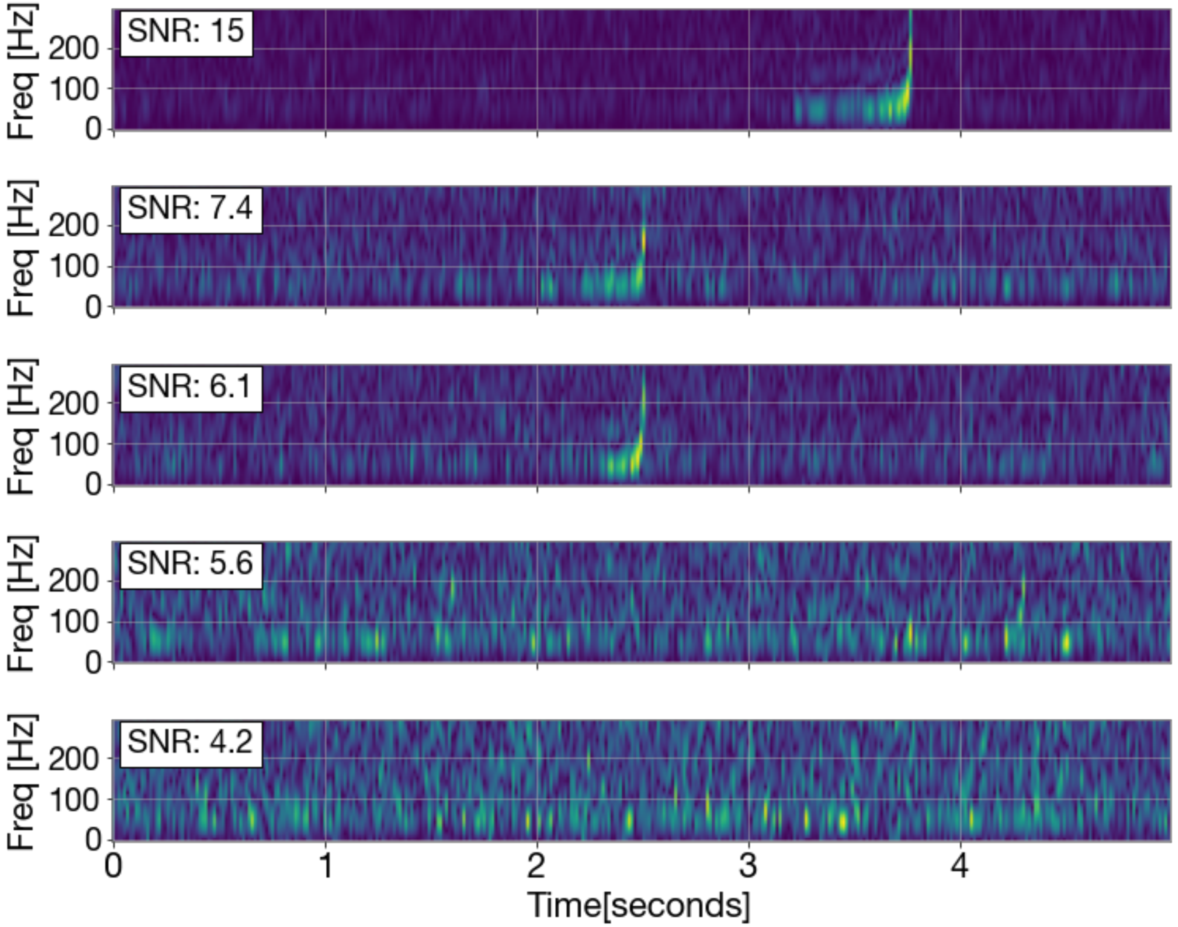
SSDD Simulation



Samples



Only-noise



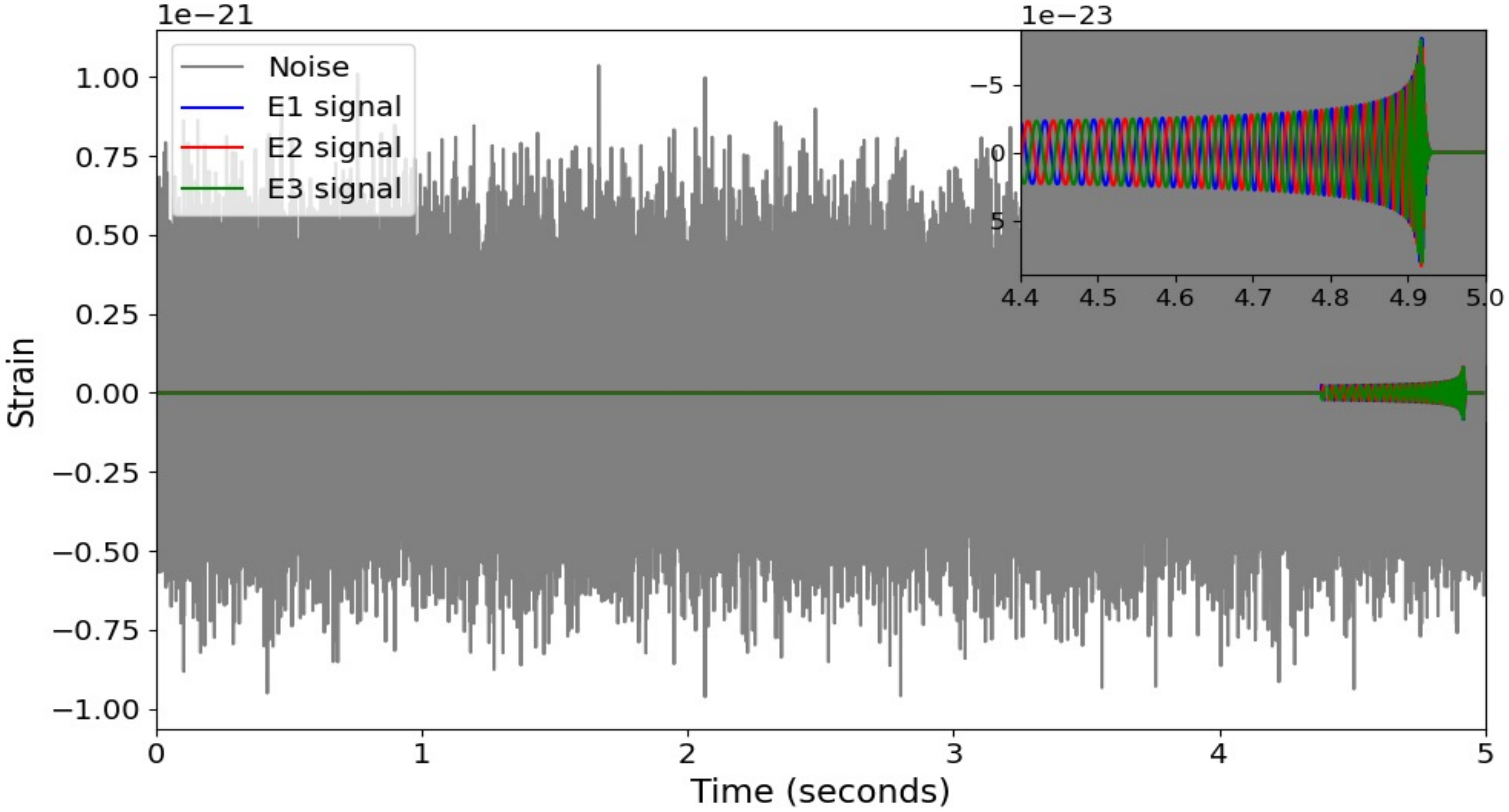
Injected (signals plus noise)

TSDCD Simulation

$$d_1(t) = h_{E1}(t) + n_1(t)$$
$$d_2(t) = h_{E2}(t) + n_2(t)$$
$$d_3(t) = h_{E3}(t) + n_3(t)$$
$$d_n(t) = n_1(t) + n_2(t) + n_3(t)$$

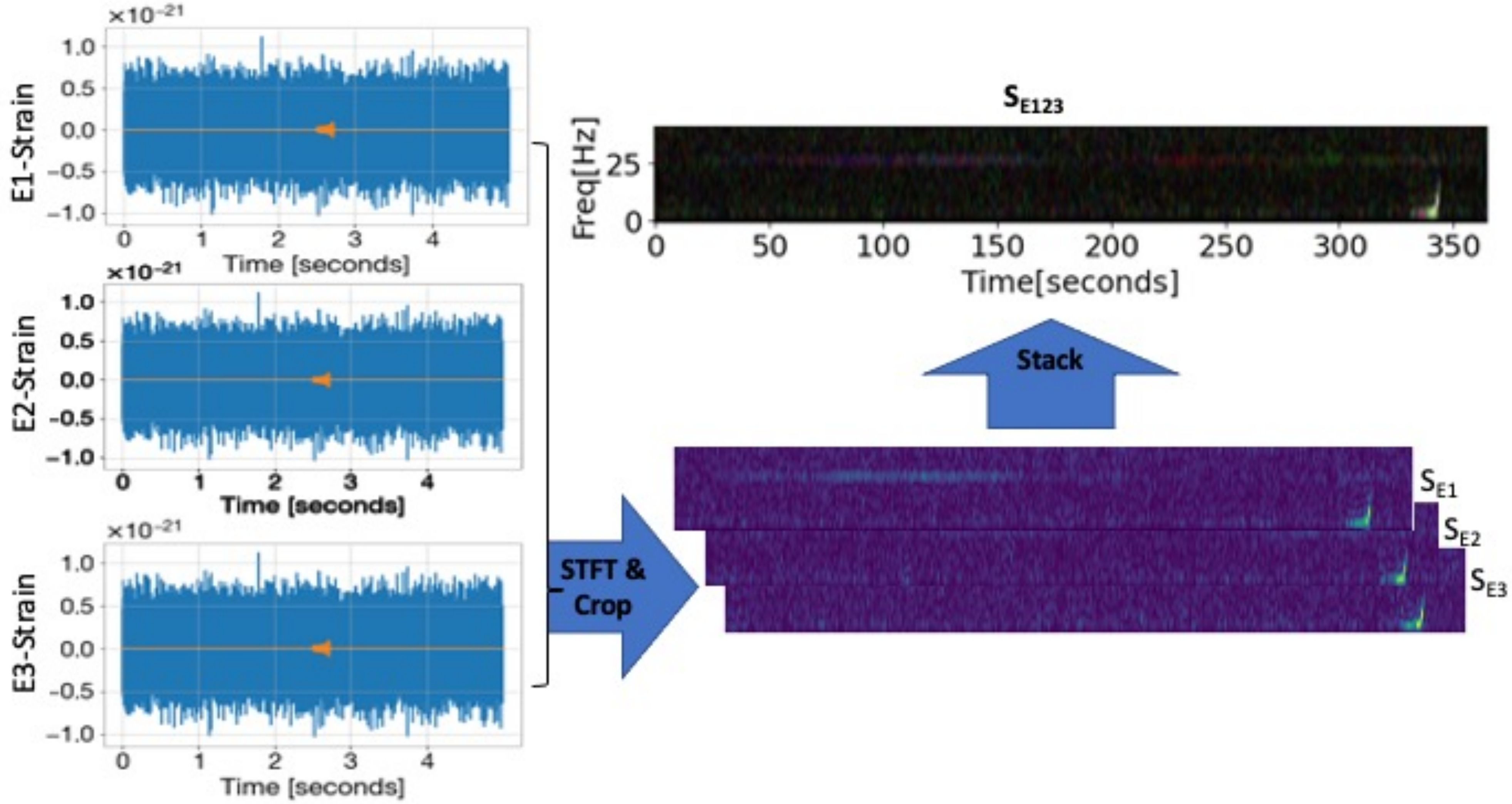
- 125,000 BBHs injections.
- f_{low} : 5 Hz, 10 Hz, 15 Hz, 20 Hz and 30 Hz.
- SNR ranges: 4-5, 5-6, 6-7, 7-8 and >8.

TSDCD Simulation



BBH: M1 = 21 | M2 = 25 | avg SNR: 7.6

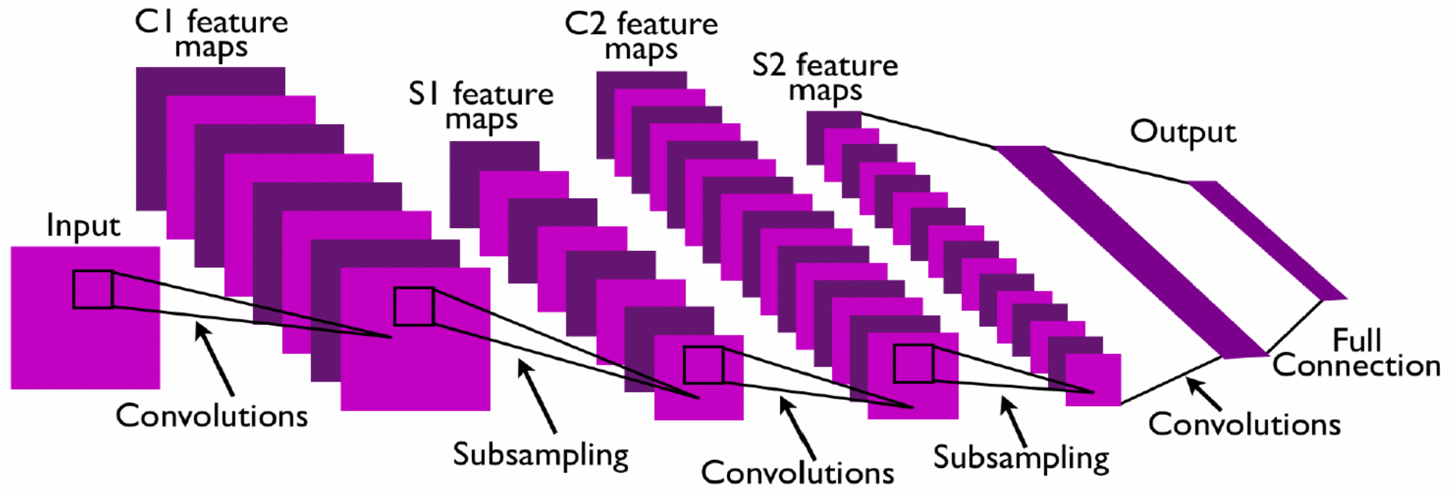
TSDCD Simulation



Spectrograms

Convolutional Neural networks (CNN)

- A type of feed-forward neural network model -- meaning the output from one layer is used as input to the next layer -- for deep learning.
- **Core Components**: Convolutional layers (feature extraction), pooling layers (downsampling), and fully connected layers (classification).
- **Key Operation**: Uses filters (kernels) to detect patterns (edges, textures, etc.) and builds complexity layer by layer.
- **Advantages**:
  - Have shown stat-of-the-art performance on image classification and object detection.
  - Automatic extraction of features unlike traditional machine learning algorithms which need to be extracted manually.
  - Ability to handle chaotic data.



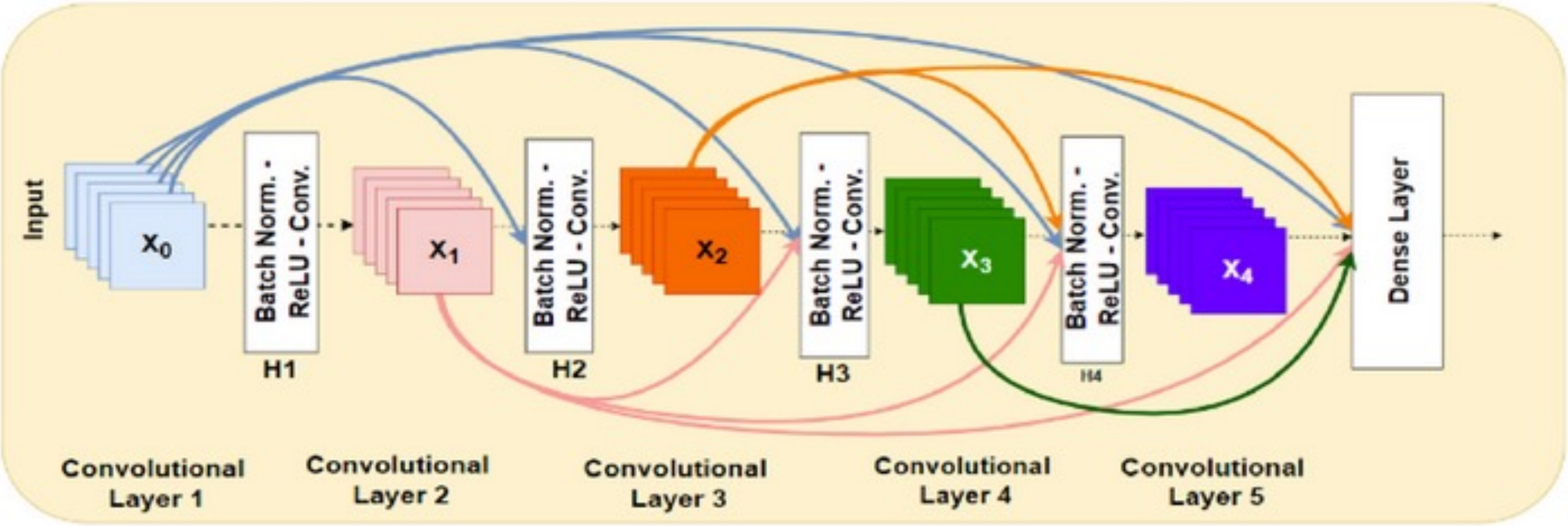
Visual Geometry Group Neural Network (VGG)

- A CNN architecture designed for image classification, known for its simplicity and effectiveness.
- **Core Idea**: Uses small (3x3) convolutional filters and increases depth with 16–19 layers for hierarchical feature extraction.
- **Key Features**: Stacked convolutional layers followed by max-pooling, and fully connected layers for classification.
- **Advantages**: Simple design, improved accuracy with deeper networks, and a balance between computational cost and performance.

VGG 16 architecture consisting of total 16 layers.

Dense Neural Network (DenseNet)

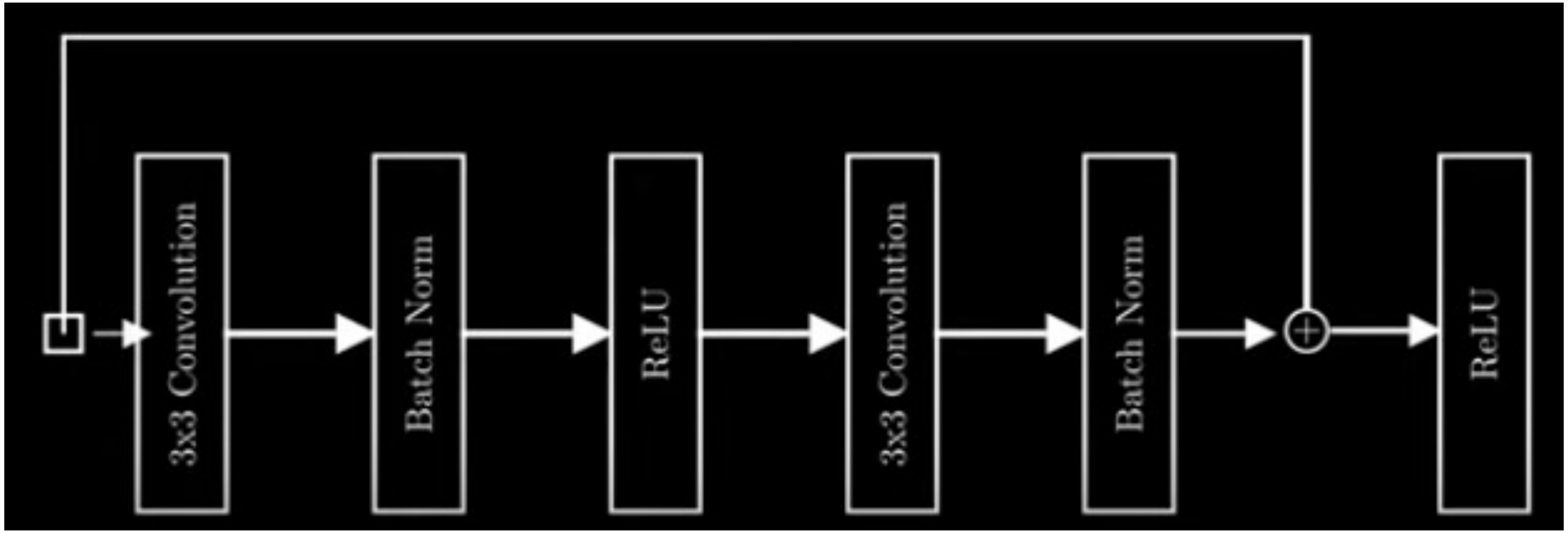
- A CNN architecture where each layer is connected to every other layer in a feed-forward manner.
- **Core Idea**: Promotes feature reuse by concatenating feature maps from all previous layers, reducing redundancy and improving efficiency.
- **Key Features**: Direct connections between layers, compact architecture with fewer parameters, and reduced risk of vanishing gradients.
- **Advantages**: Efficient use of parameters, improved flow of gradients, and better performance with fewer computations compared to traditional CNNs.



DenseNet architecture consisting of five concatenated convolutional layers.

Deep Residual Network (ResNet)

- A deep CNN architecture introducing *residual connections* to ease the training of very deep networks.
- **Core Idea**: Instead of learning a direct mapping, ResNet learns residuals (differences) by adding shortcut connections between layers.
- **Key Features**: Shortcut (skip) connections bypass some layers, allowing gradients to flow more effectively and mitigating vanishing gradient problems.
- **Advantages**: Enables the training of extremely deep networks, improves accuracy, and reduces degradation issues as networks grow deeper.



Residual block

Evaluation Metrics

$$\text{precision} = \frac{TP}{TP+FP}$$

TP: true positives, FP: false positives (FP), FN: false negatives

- ✓ Measures the proportion of the positive predictions.

$$\text{F1score} = 2 * \frac{precision * recall}{precision + recall}$$

- ✓ provides a balanced measure of a classifier's performance by considering both **Precision** and **Recall**.
- ✓ helps compare and evaluate the overall performance of classification models.

$$\text{recall} = \frac{TP}{TP+FN}$$

- ✓ Measures the ability of a model to correctly identify all positive cases.

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP+TN}$$

- ✓ Quantifies how often the model incorrectly classifies injected samples as noise.

The higher value of recall, precision and F1-score, the better performance of the model.

SSDD Experiment

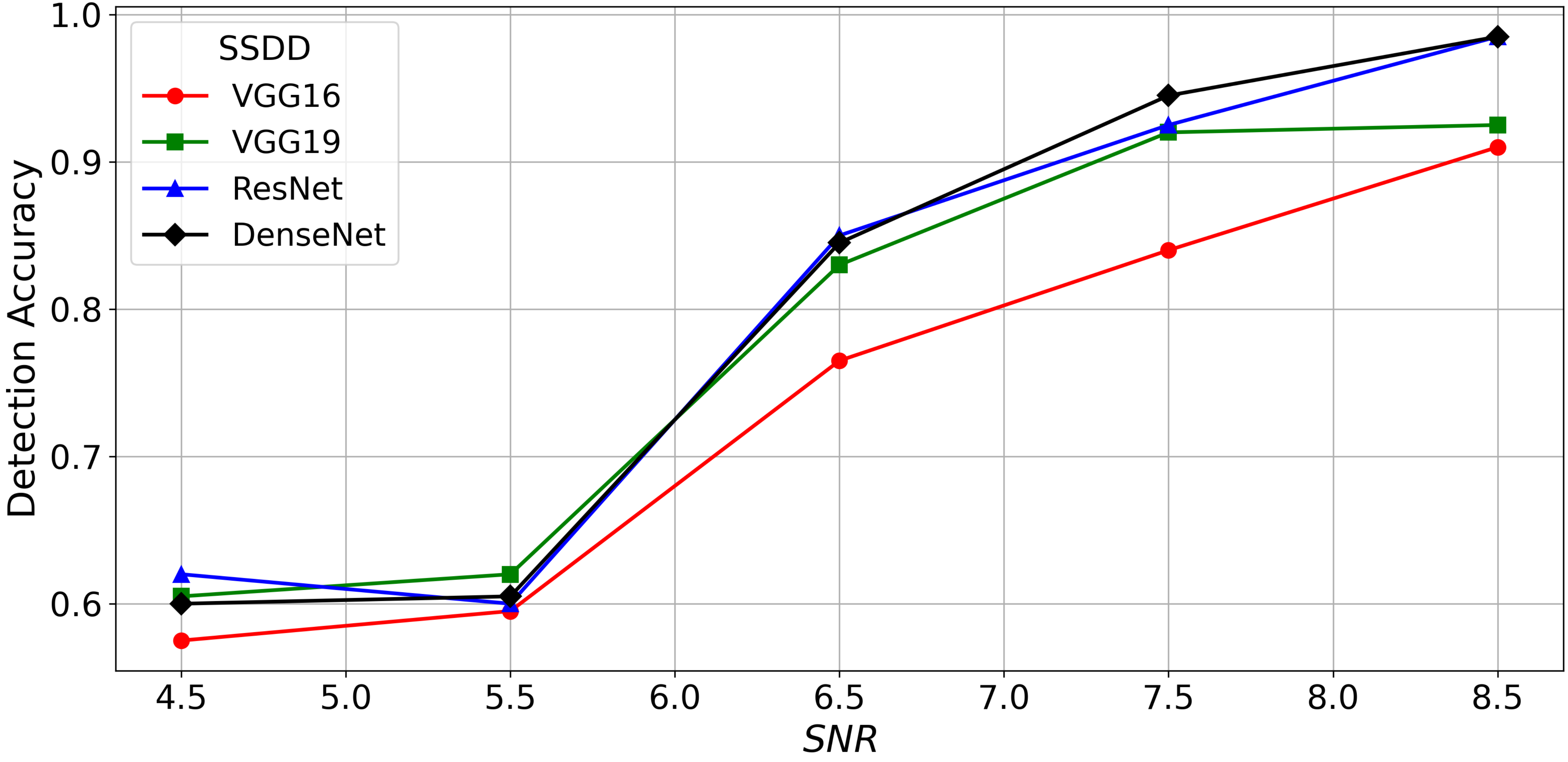
CNN models: VGG 16, VGG 19, RestNet-101, DenseNet-121

- A batch size: 256, learning rate: 0.0001, epochs: 200
- Root Mean Square Propagation (RMSpro) optimizer.
- Input layer shape: 365 × 42 × 1

Type	Number of Sample	Train	Test	Val
Injected	25,000	17,000	4000	4000
Only noise	25,000	17,000	4000	4000
Total	50,000	34,000	8000	8000

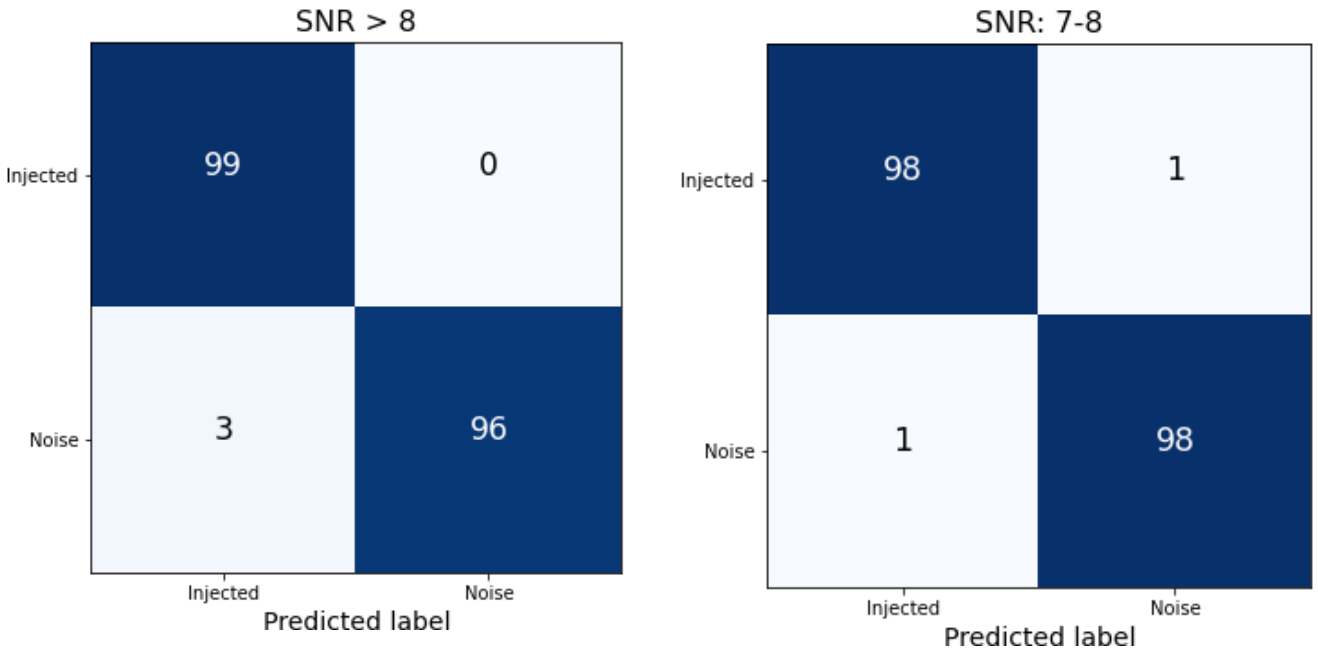
Total number of: injected and only noise spectrograms for training, testing and validation.

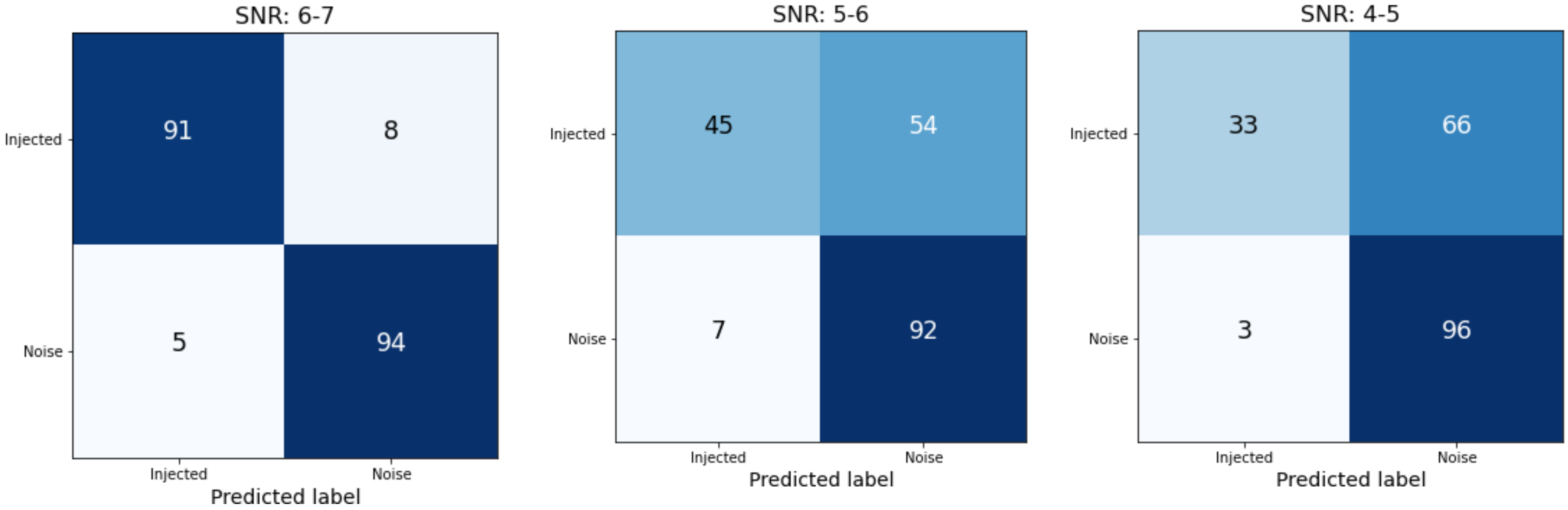
SSDD Results



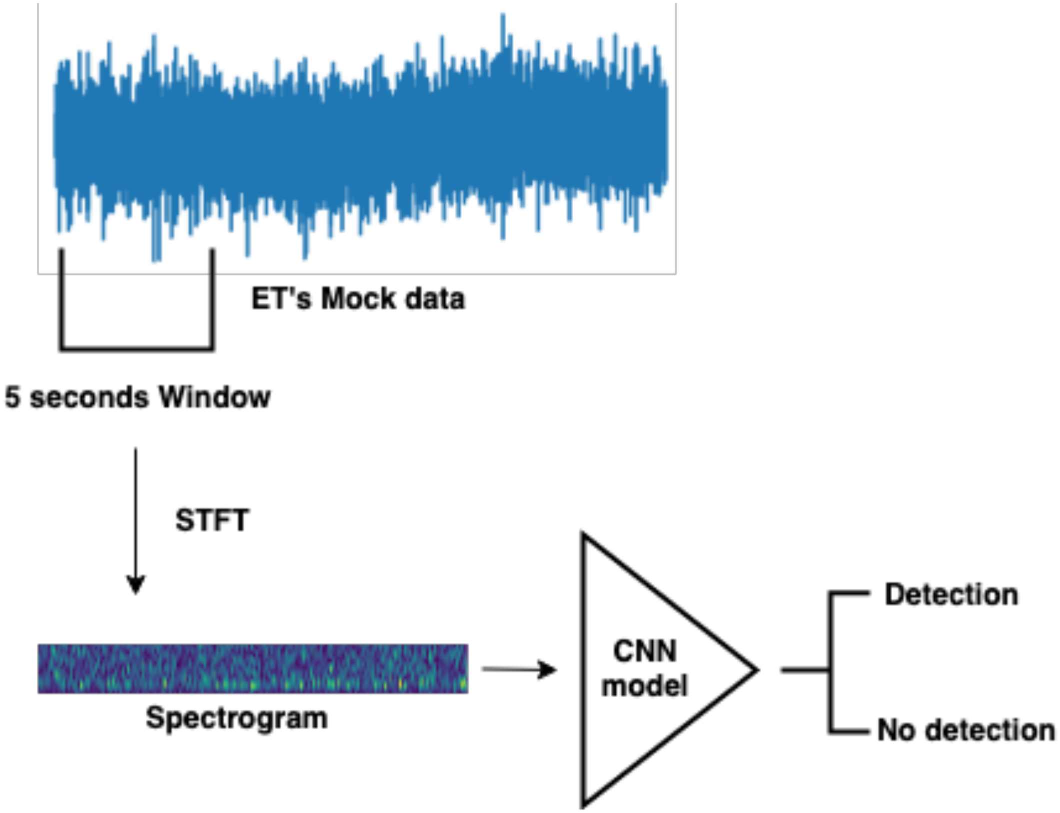
Alhassan et al. (2023), MNRAS

SSDD Results





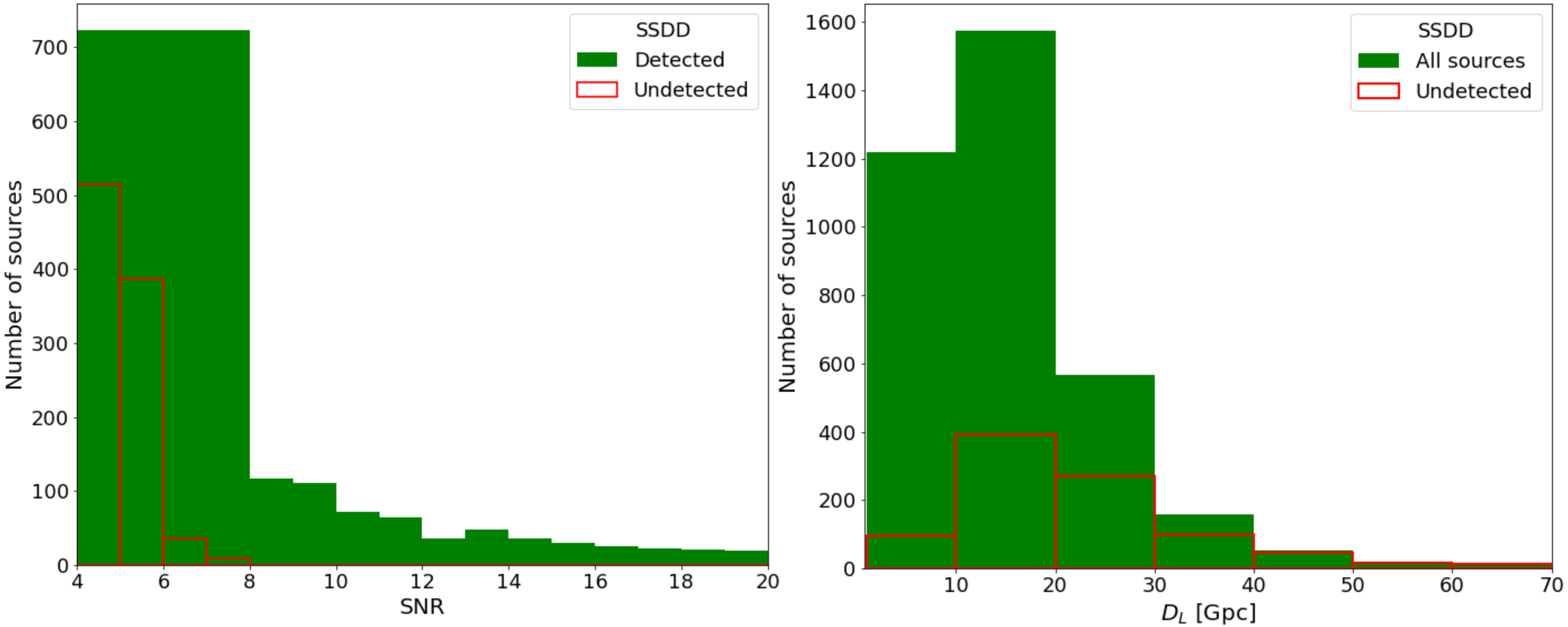
SSDD ≈ 6 hours mock



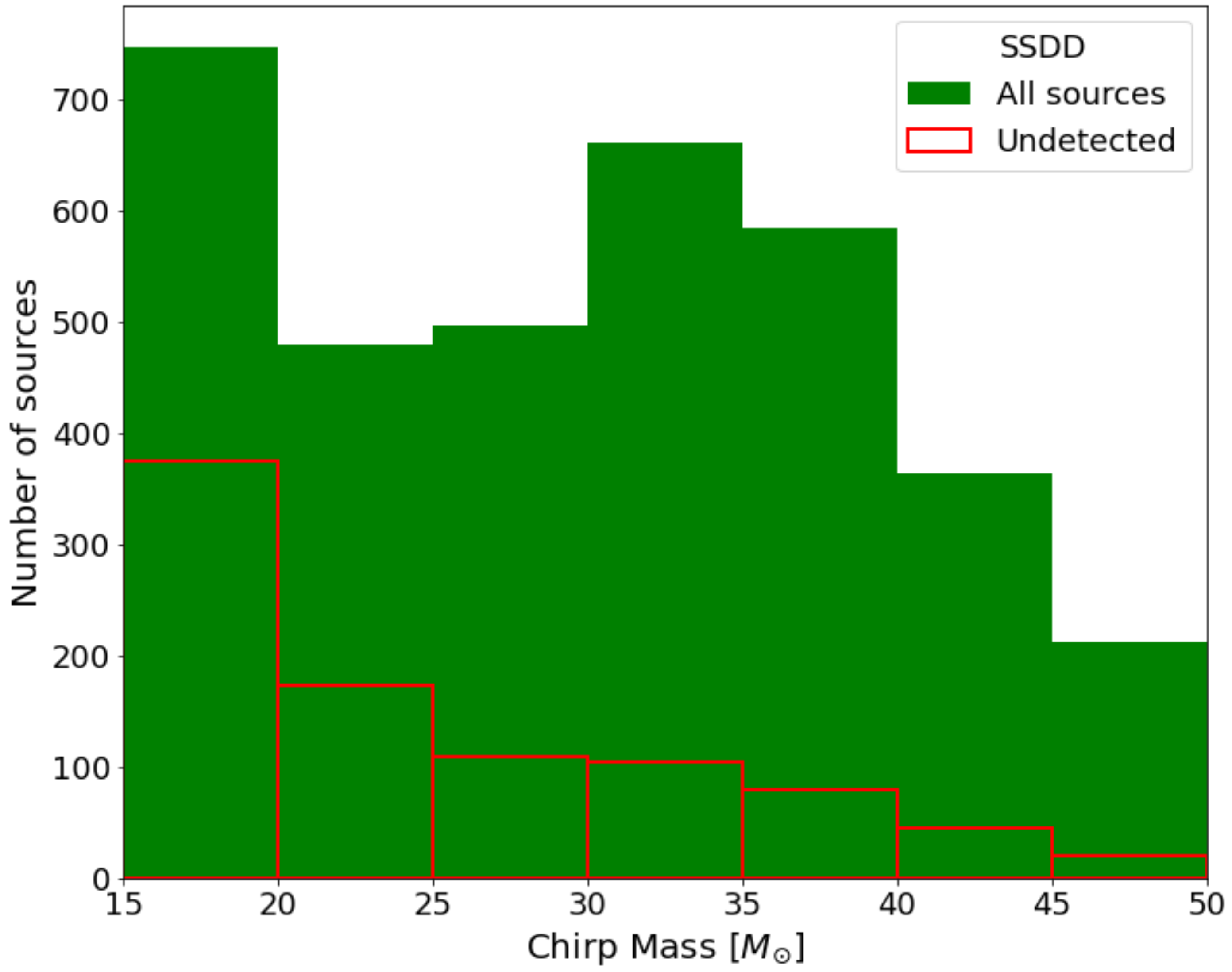
Infernencing on SSDD mock

Hours	SNR range	# of injected sources	# of detection	FPR/FNR
1st	> 8	723	722	0.001
2nd	7-8	723	713	0.014
3rd	6-7	723	687	0.050
4th	5-6	723	336	0.535
5th	4-5	723	208	0.712
6th	Noise only	0	24	0.003

SSDD Mock results



SSDD Mock results



TSDCD Experiment

CNN models: ResNet-101

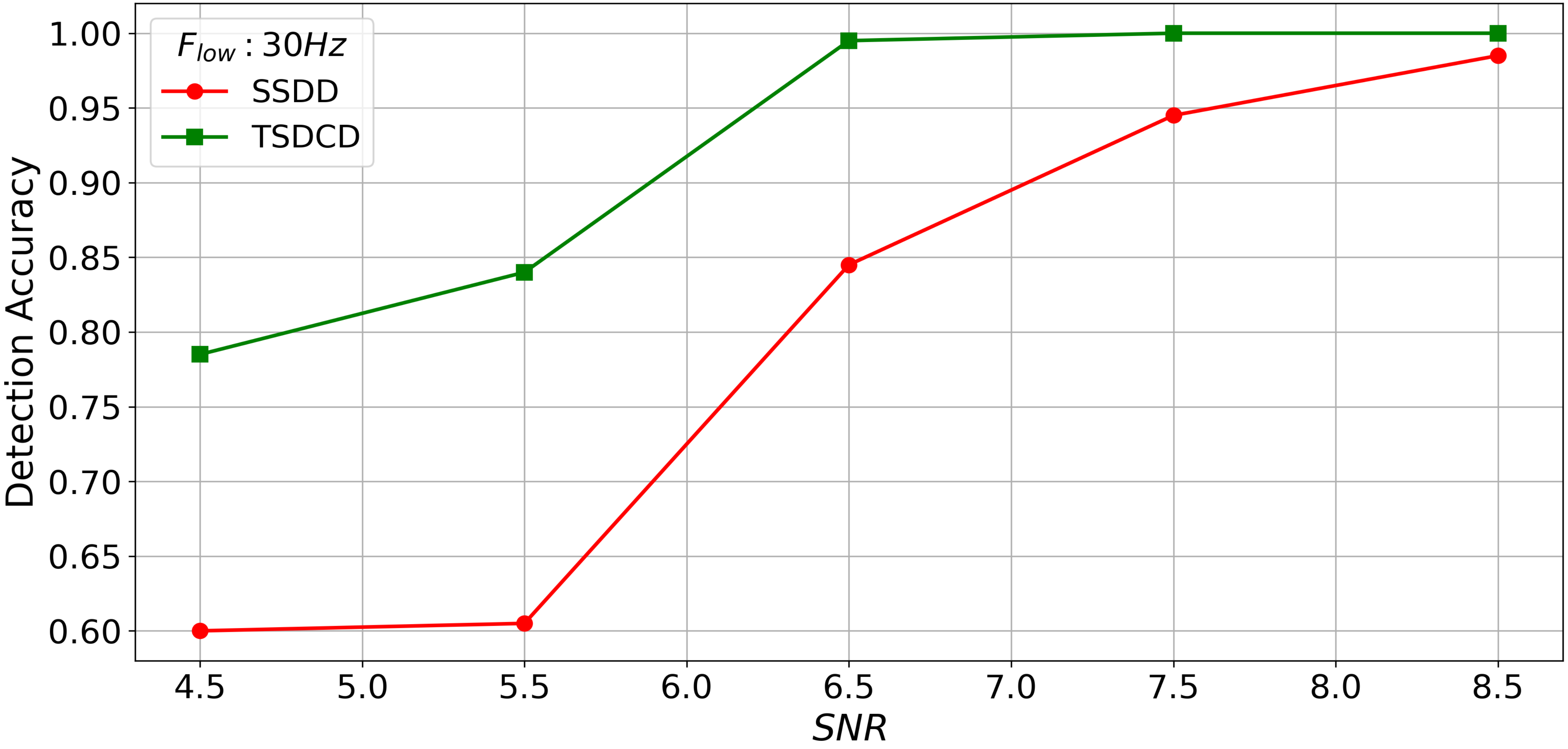
- A batch size: 256, learning rate: 0.0001, epochs: 200
- Root Mean Square Propagation (RMSpro) optimizer.
- Input layer shape: 365 × 42 × **3**

Type	Number of Sample	Train	Test	Val
Injected	125,000	85,000	20,000	20,000
Only noise	125,000	85,000	20,000	20,000
Total	250,000	170,000	40,000	40,000

Classification Report

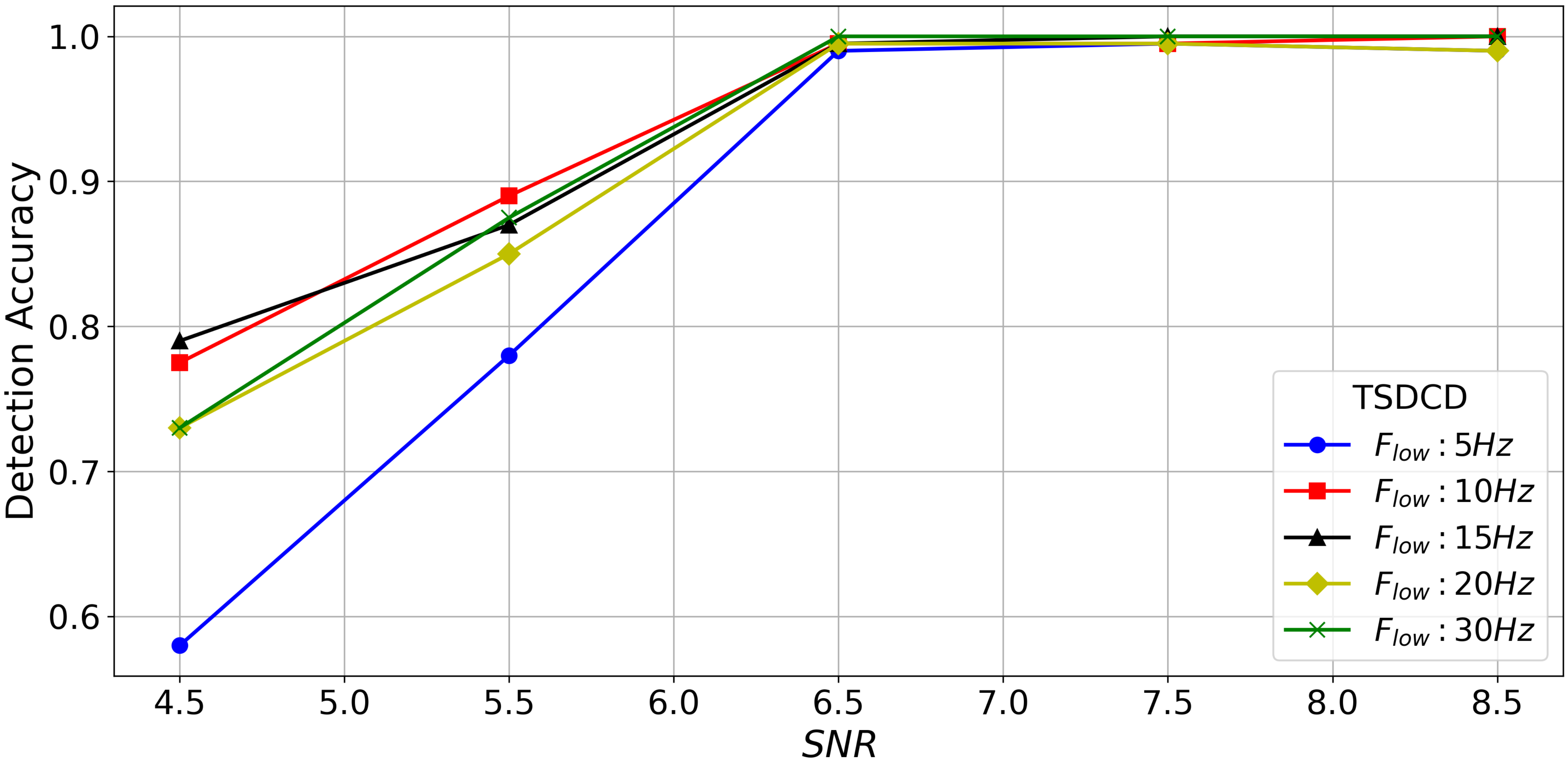
Type	precision	recall	f1-score	support
Injected	0.993	0.818	0.897	20,000
Only noise	0.845	0.994	0.914	20,000
avg/ total	0.919	0.906	0.905	40,000

TSDCD Results



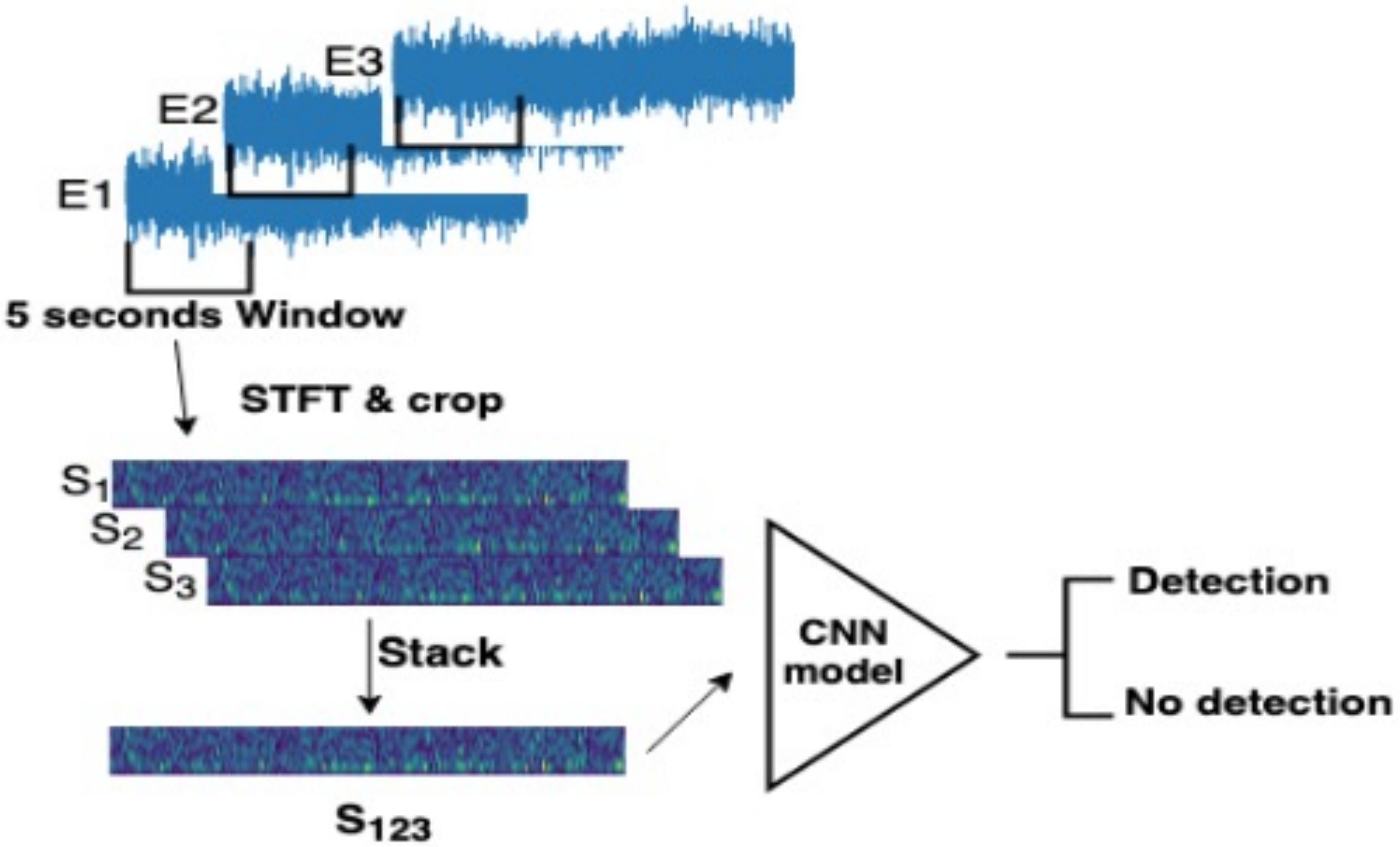
SSDD (Alhassan et al. (2022)) versus TSDCD for sources with $Flow$ of 30 Hz.

TSDCD Results



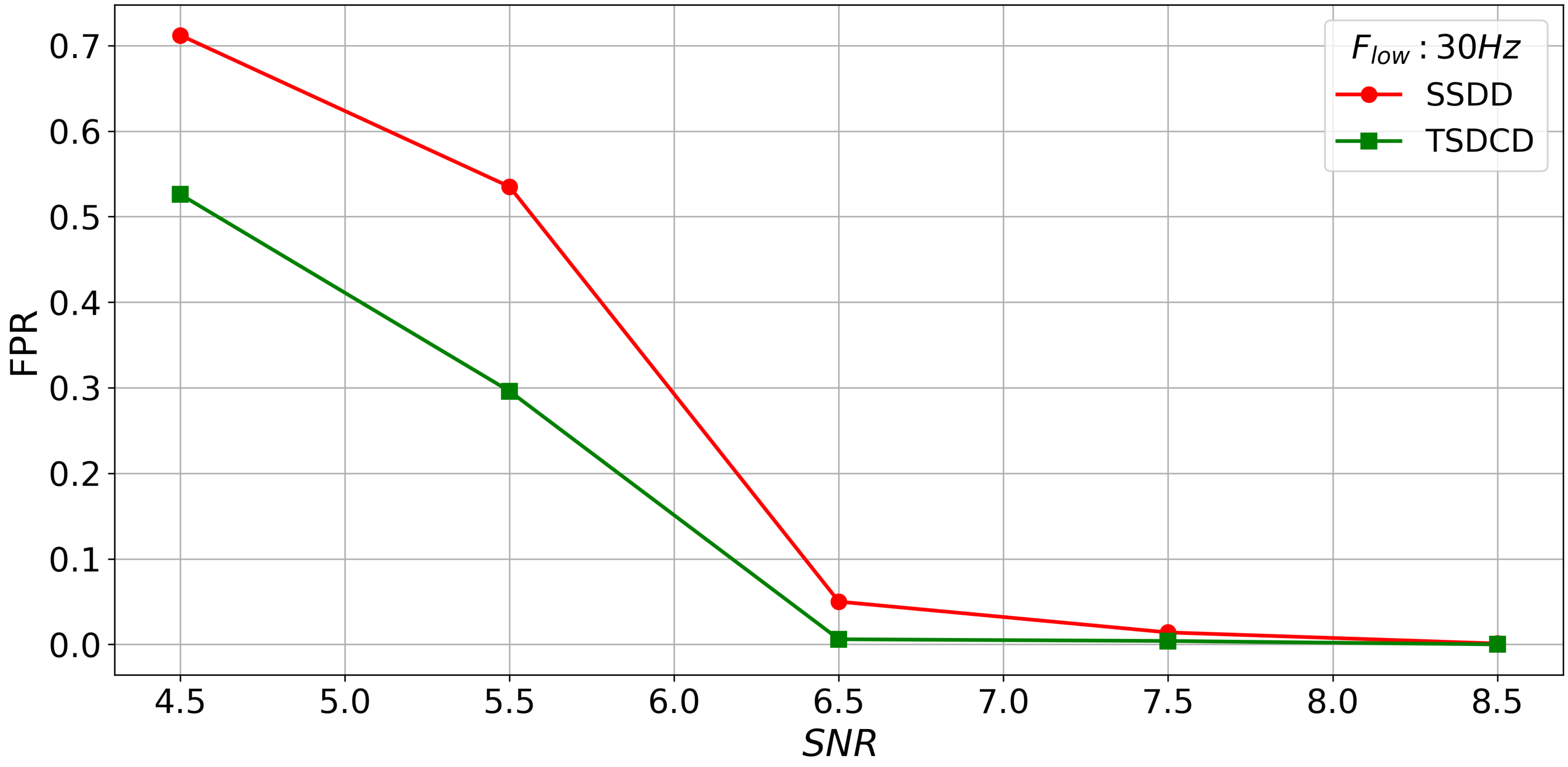
TSDCD for sources with F_{low} of 5 Hz, 10 Hz, 15 Hz, 20 Hz and 30 Hz.

TSDCD ≈25 hours mock data

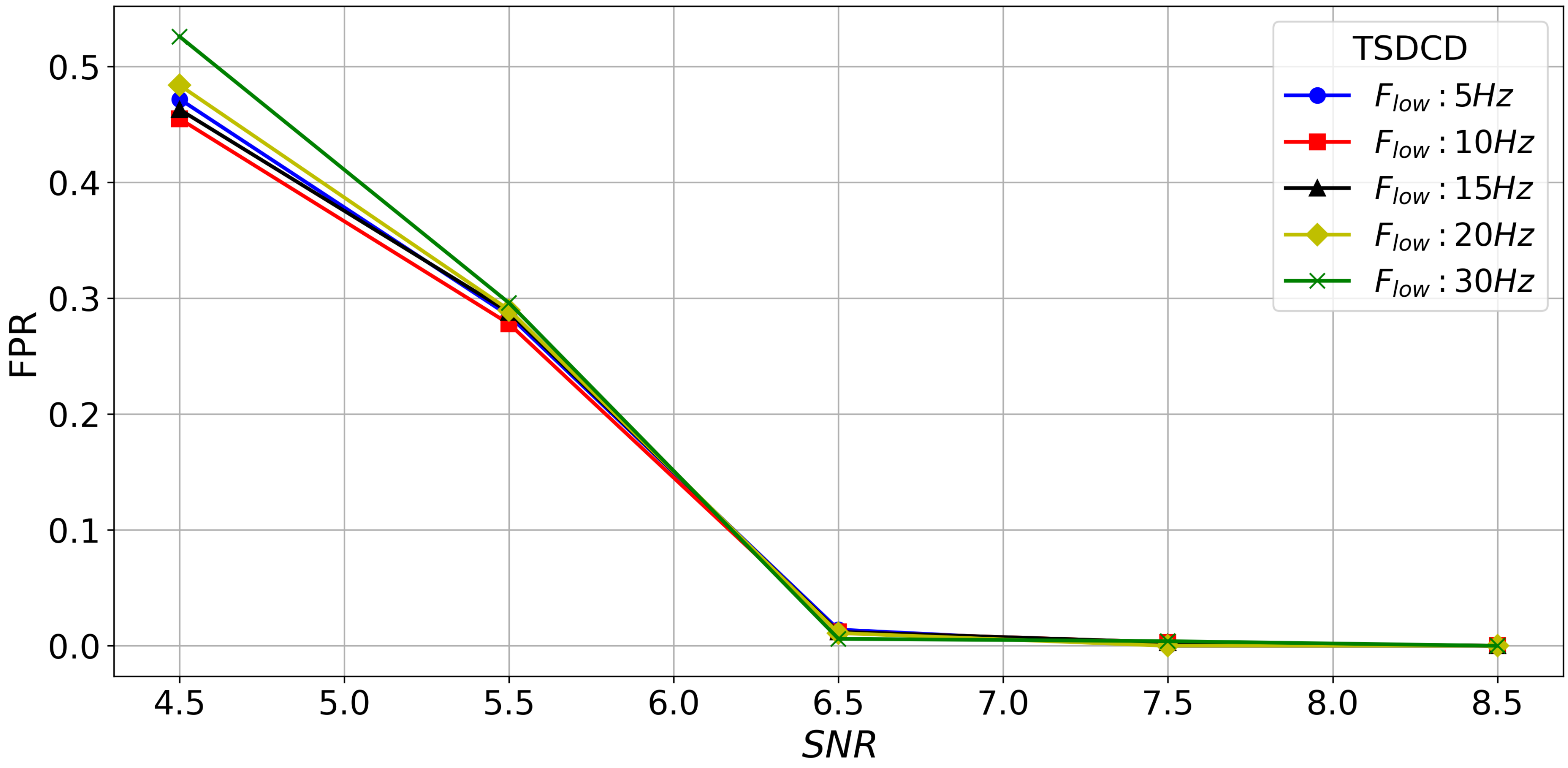


Infernencing on TSDCD mock

TSDCD Mock Results

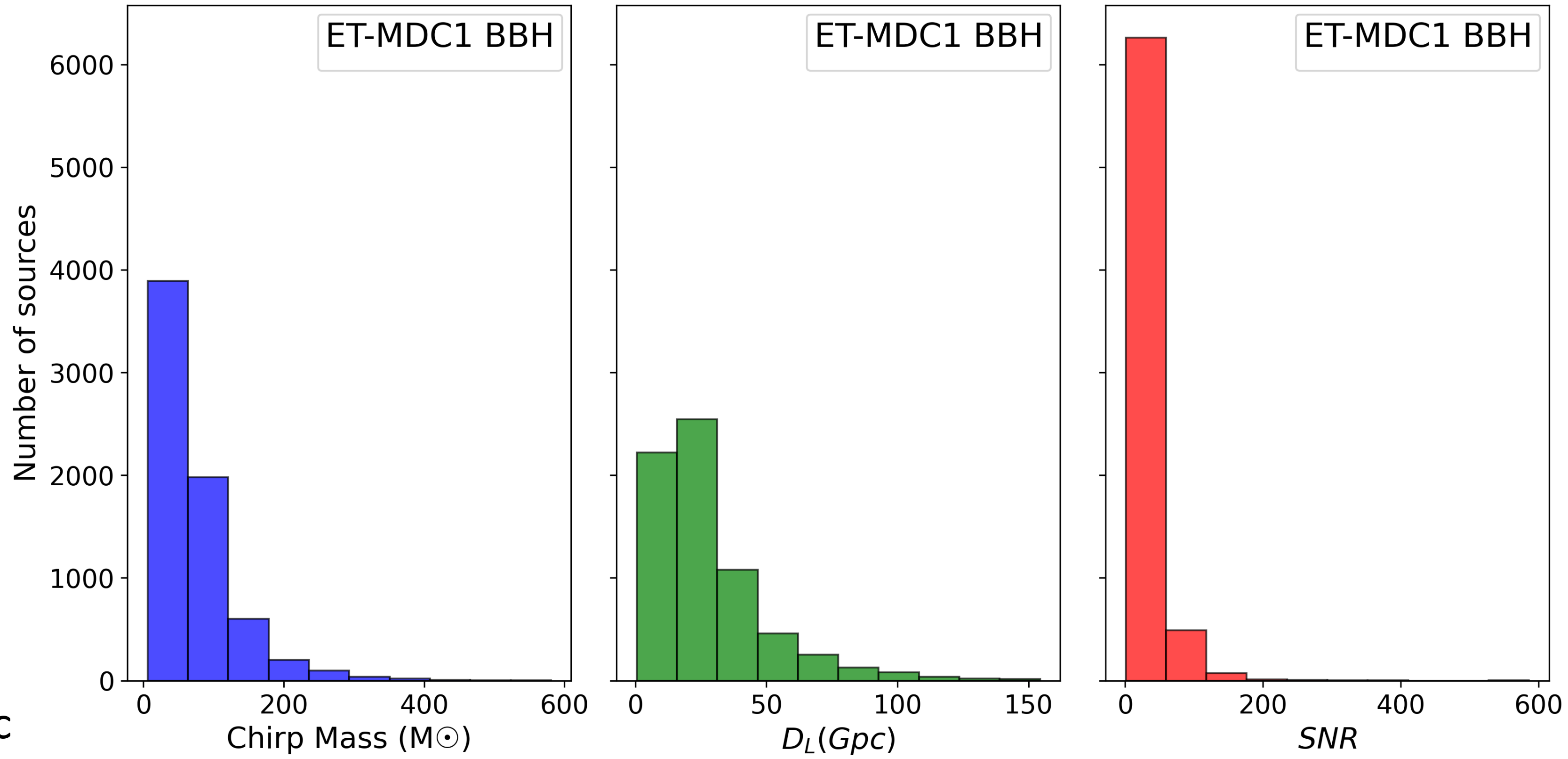


TSDCD Mock Results

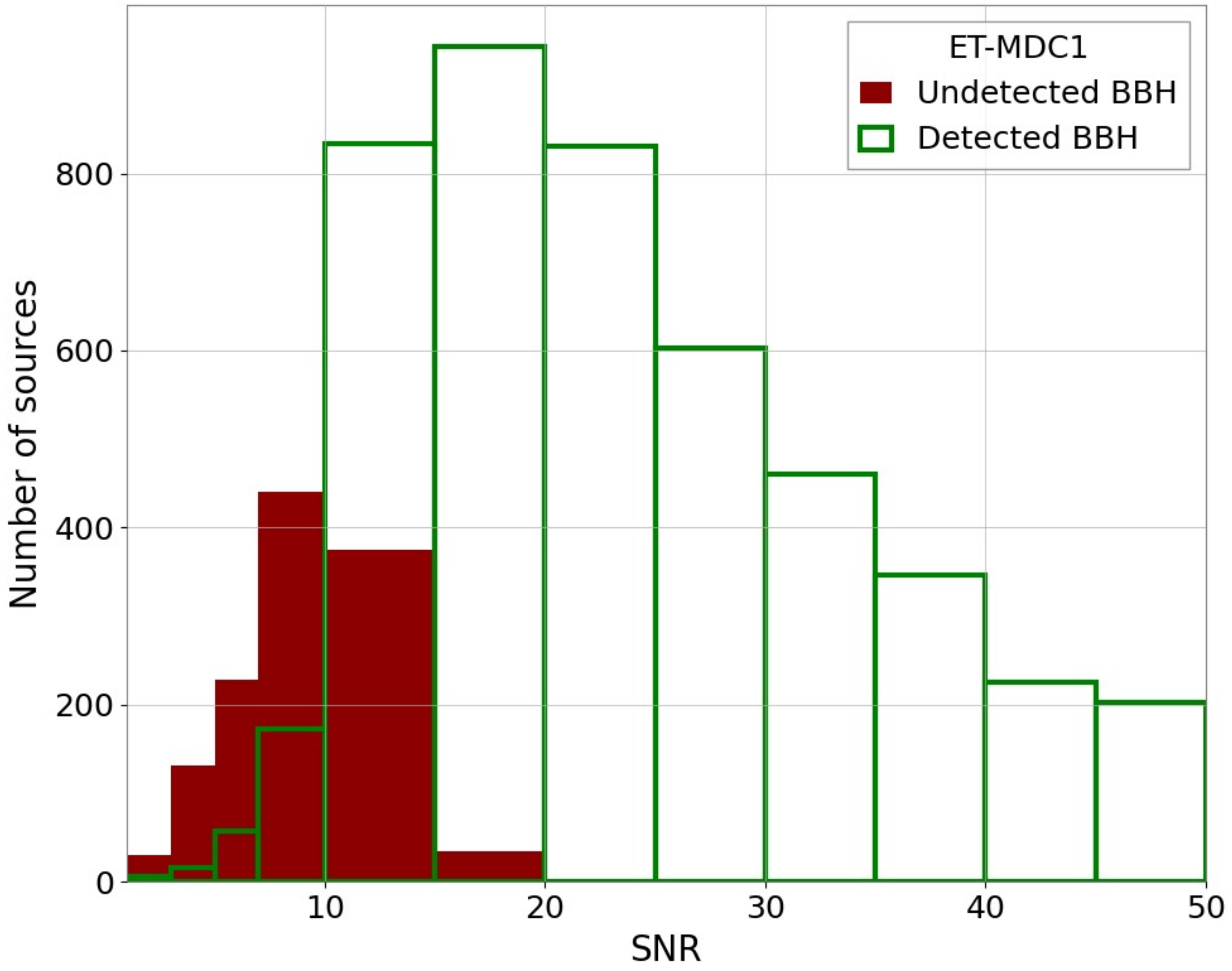


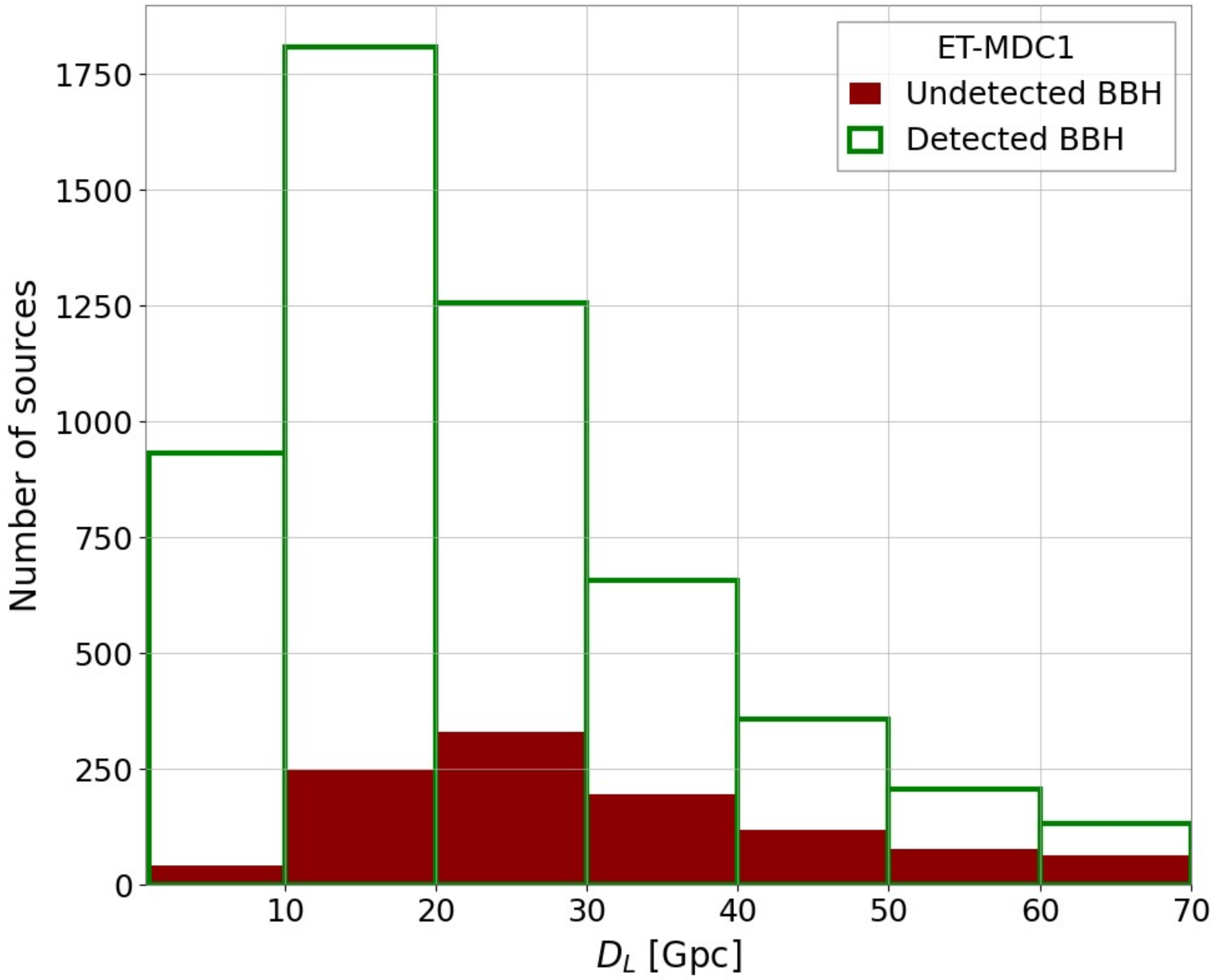
ET-MDC1: Einstein Telescope mock Data Challenge

- ➢ First released in 2012 and updated in early 2024.
- ➢ 30.8 days (1.3 Terabytes).
- ➢ Contains a continuous GW signal plus noise, noise only and null stream.
- ➢ Overlapping signals.
- ➢ Sampled at 8192 Hz.
- ➢ 59,540 BNS | 6,578 BBH | 1,977 BHNS
- ➢ Optimal SNR ranges between 0.13 and 586.12.
- ➢ D_L of BBH systems ranges from 0.5 Gpc to 154.37 Gpc.

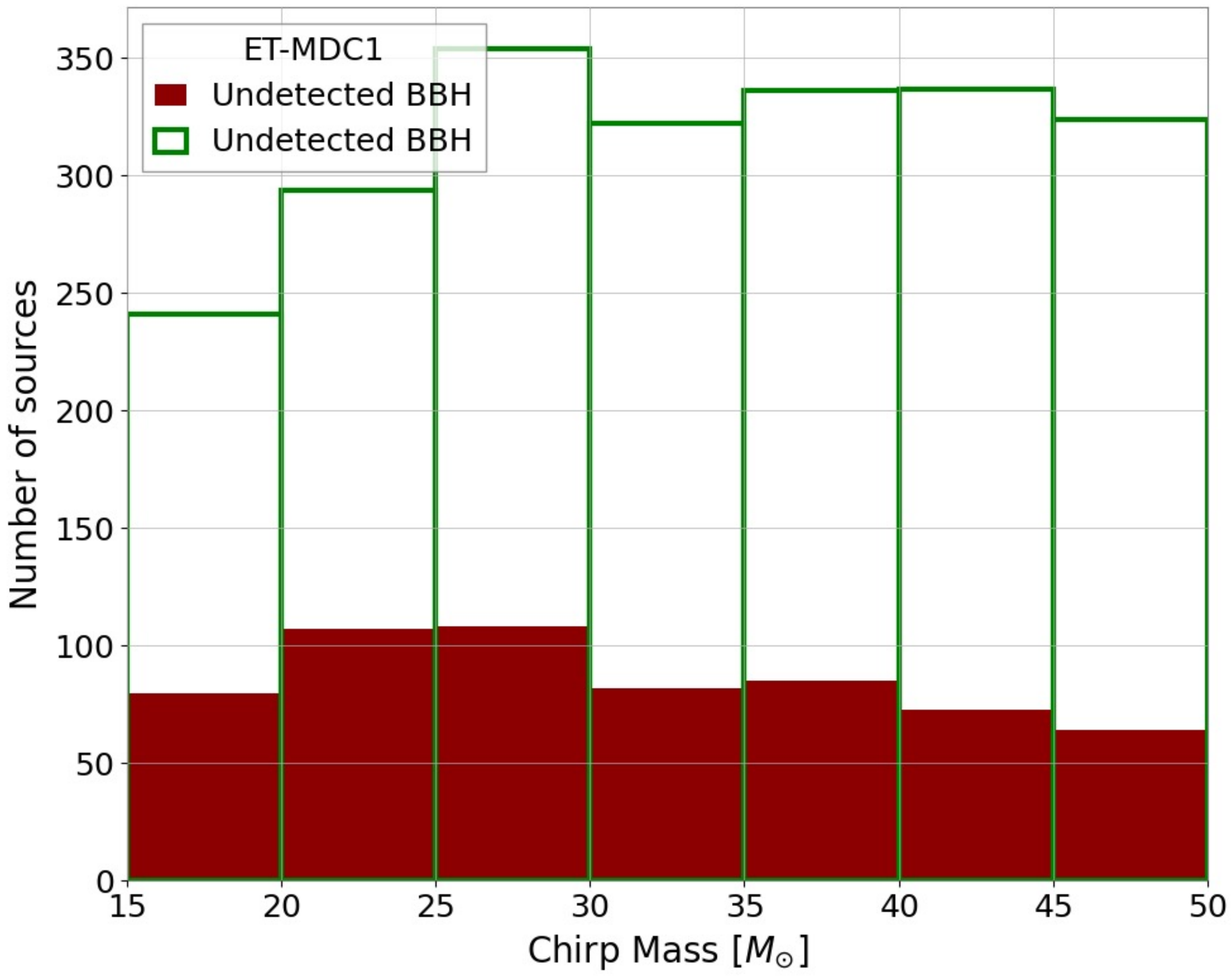


ET-MDC1 Results

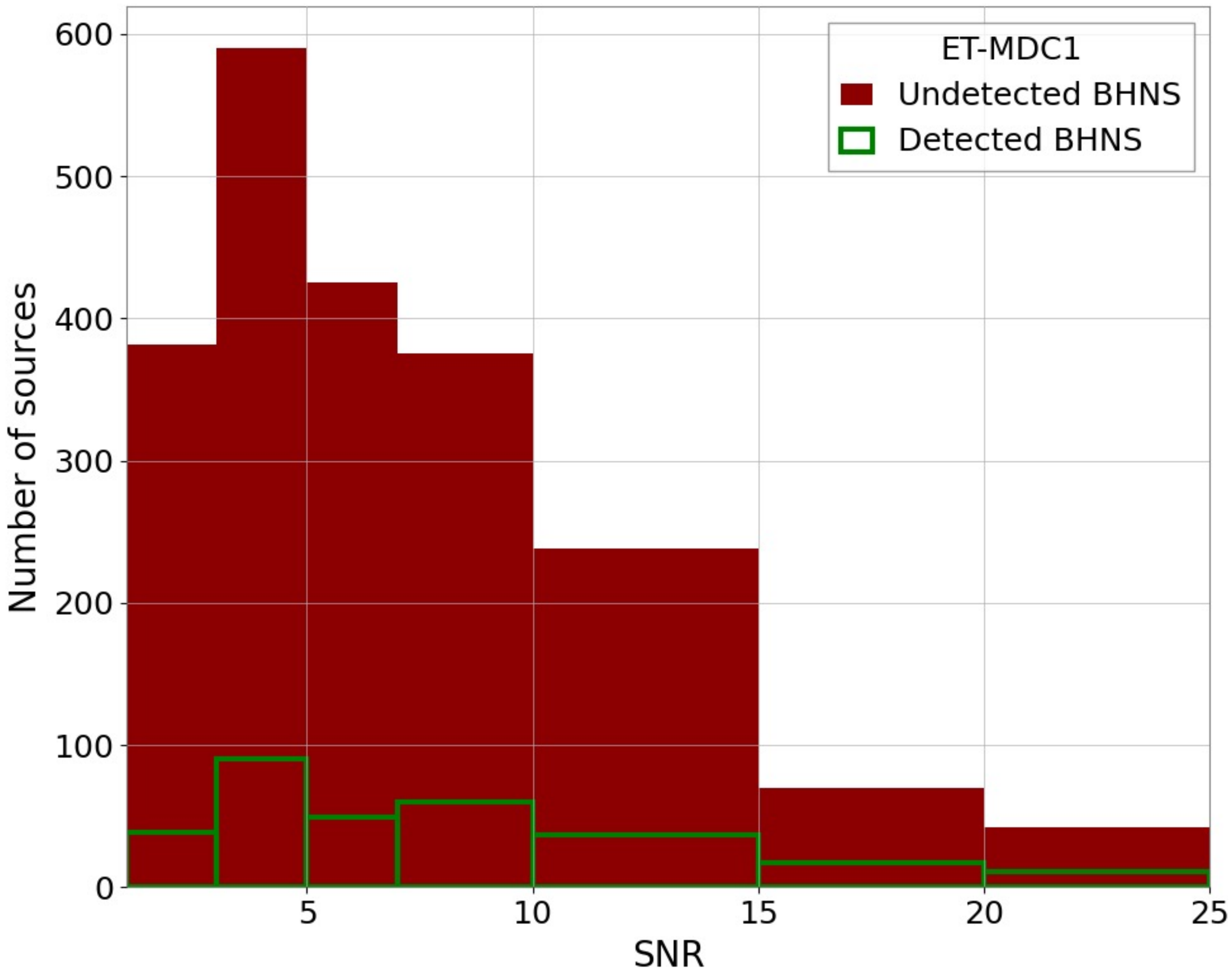


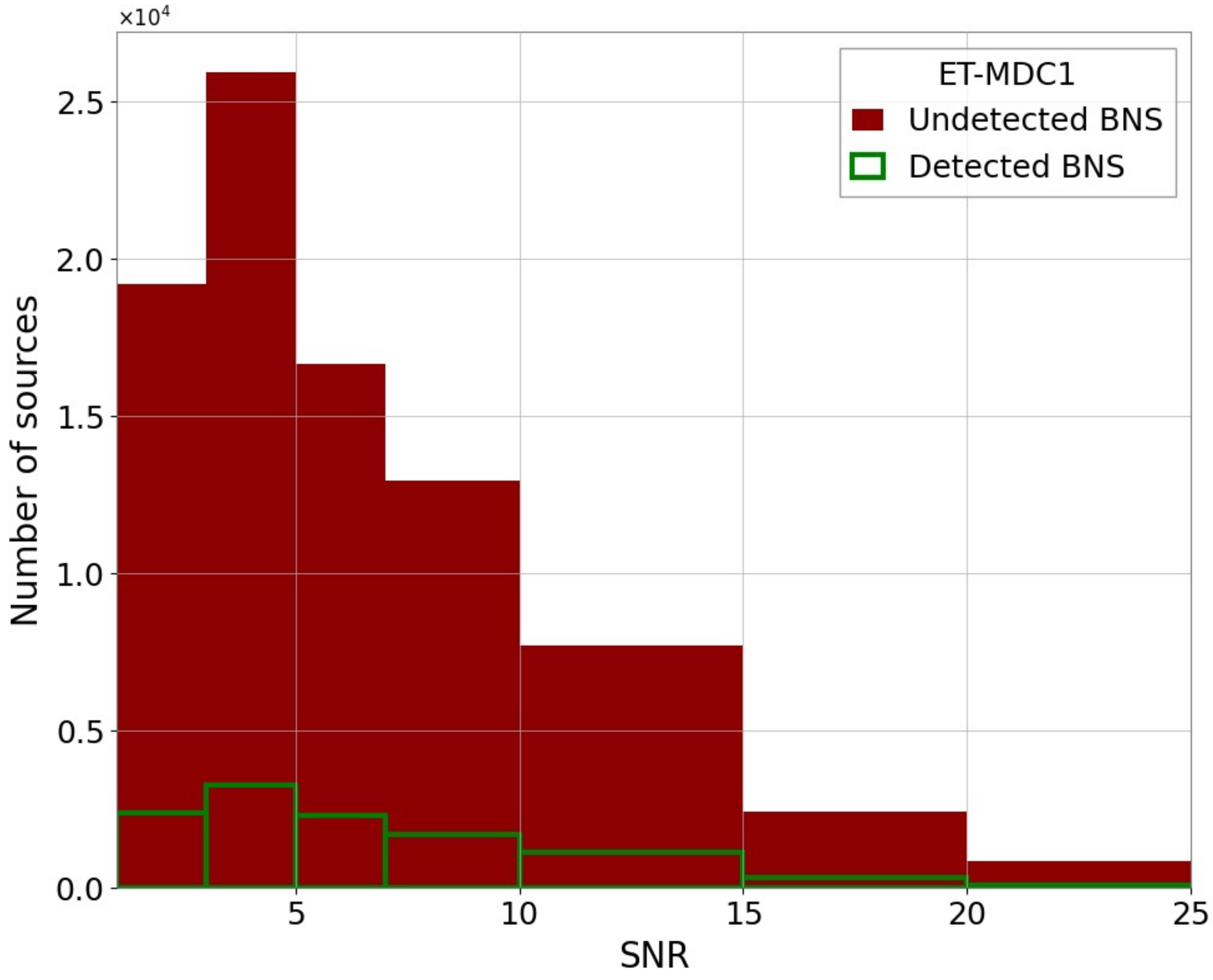


ET-MDC1 Results



ET-MDC1 Results





- ✓ 11,477 BNS mergers (with optimal SNR starting from 0.2).
- ✓ 323 BHNS mergers (with optimal SNR starting from 0.1).

MDC1 – Null and only noise

Threshold	Null	Noise
0.5	0	10
0.3	0	1
0.1	0	0

FNR from one week of noise and null data only

PyMerger

- PyMerger is a Python tool for detecting BBH mergers from ET, built based on our trained ResNet model.
- The current version handles only gravitational wave frame file format (.gwf)
- 1.9 minutes to scan one hour of data on an average laptop without GPUs.

https://github.com/wathela/PyMerger

```
pip install PyMergers
```

Installation

1. Clone the repository:

```
git clone https://github.com/your-username/PyMerger.git
cd PyMerger
```

2. Install the required Python packages:

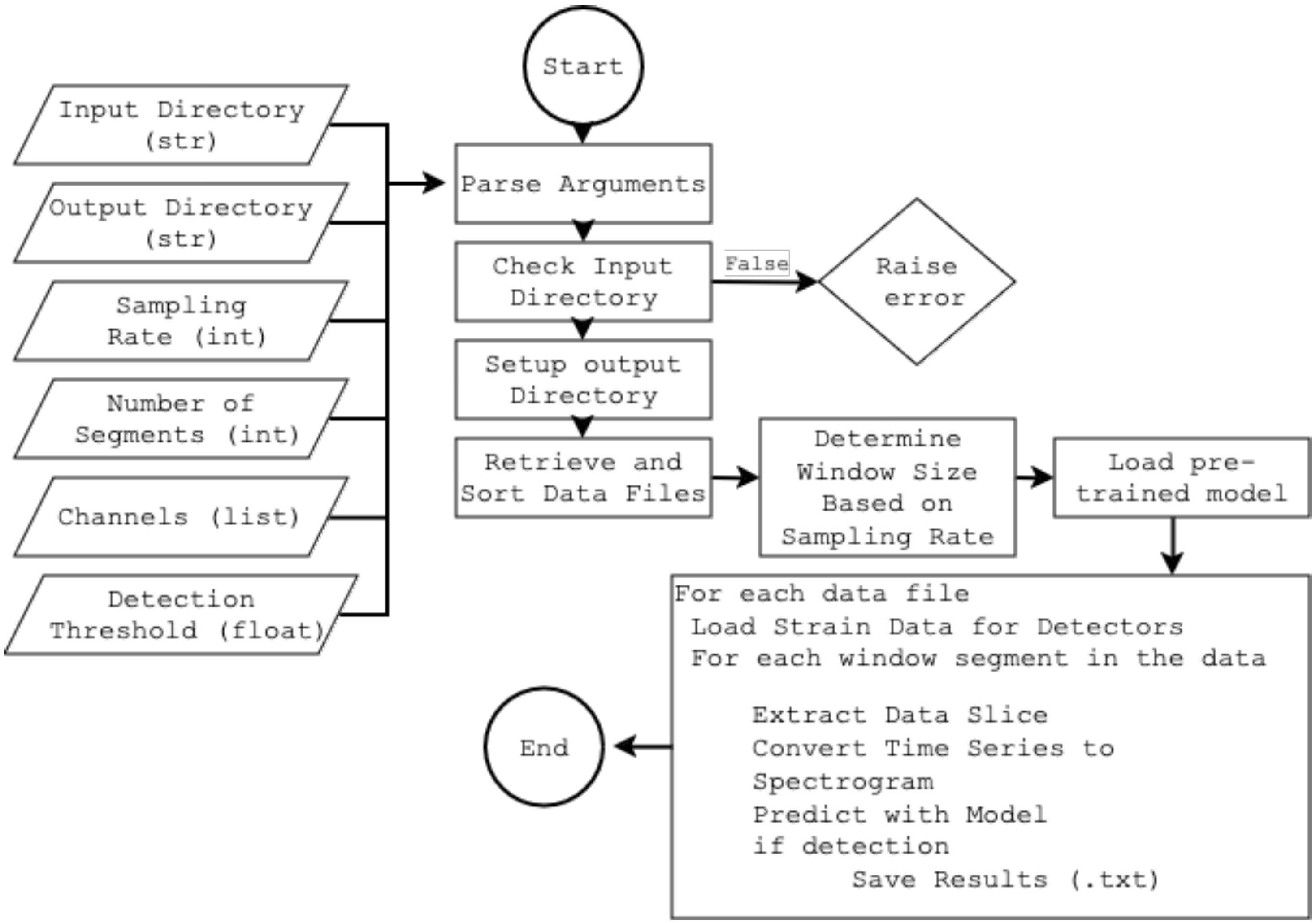
```
pip install -r requirements.txt
```

Usage

PyMerger assumes that each sub-detector of ET will have a separate .gwf file in three separate directories (E1, E2, E3). The data input path should point to the folder where these three directories are located.

```
usage: pymerger.py [-h] [-r {8192,4096}] [-n NO_SEGMENT] [-c CHANNELS CHANNELS CHANNELS] [-t T

optional arguments:
  -h, --help            show this help message and exit
  -r {8192,4096}, --sampling-rate {8192,4096}
                        Sampling rate of the input data (either 8192 or 4096). Default is 8192
  -n NO_SEGMENT, --no-segment NO_SEGMENT
                        Number of data segments to be processed for each detector (i.e., numbe
                        Files in the input directory will be sorted, and the first 'n' files u
                        Default is 1 which means there are 1 unique file from each detector.
  -c CHANNELS CHANNELS CHANNELS, --channels CHANNELS CHANNELS CHANNELS
                        List of the THREE channels to be processed. Default is ['E1:STRAIN', '
  -t THRESHOLD, --threshold THRESHOLD
                        Threshold value for merger detection. A value between 0.5 and 0, where
                        a lower false positive rate. Default is 0.1 (accepting detection with
  -i INPUT_FILE_DIR, --input-file-dir INPUT_FILE_DIR
                        Directory containing the input .gwf files.
  -f OUTPUT_DIR, --output-dir OUTPUT_DIR
                        Directory to store the results.
  --verbose             Enable verbose mode to print update messages. Default is true.
```



Start
Input Directory (str)
Output Directory (str)
Sampling Rate (int)
Number of Segments (int)
Channels (list)
Detection Threshold (float)
Parse Arguments
Check Input Directory
False
Raise error
Setup output Directory
Retrieve and Sort Data Files
Determine Window Size Based on Sampling Rate
Load pre-trained model
For each data file
Load Strain Data for Detectors
For each window segment in the data
Extract Data Slice
Convert Time Series to Spectrogram
Predict with Model
if detection
Save Results (.txt)
End

PyMerger

Output sample

```
Starting_time | End_time       | Prob |Pred
1000409750.000,1000409752.500,0.9300,0
1000409770.000,1000409772.500,0.9309,0
1000409772.500,1000409775.000,0.9213,0
1000409810.000,1000409812.500,0.9611,0
1000409860.000,1000409862.500,0.9515,0
1000409935.000,1000409937.500,0.9849,0
1000409995.000,1000409997.500,0.9000,0
1000410077.500,1000410080.000,0.9019,0
1000410095.000,1000410097.500,0.9132,0
1000410170.000,1000410172.500,0.9226,0
1000410180.000,1000410182.500,0.9108,0
1000410187.500,1000410190.000,0.9043,0
1000410250.000,1000410252.500,0.9480,0
1000410350.000,1000410352.500,0.9053,0
1000410425.000,1000410427.500,0.9621,0
1000410540.000,1000410542.500,0.9075,0
1000410555.000,1000410557.500,0.9915,0
1000410565.000,1000410567.500,0.9673,0
1000410612.500,1000410615.000,0.9934,0
1000410655.000,1000410657.500,0.9236,0
1000410680.000,1000410682.500,0.9430,0
1000410745.000,1000410747.500,0.9950,0
1000410822.500,1000410825.000,0.9225,0
1000410860.000,1000410862.500,0.9000,0
1000410900.000,1000410902.500,0.9311,0
1000410912.500,1000410915.000,0.9312,0
1000411030.000,1000411032.500,0.9000,0
1000411085.000,1000411087.500,0.9490,0
1000411122.500,1000411125.000,0.9100,0
1000411170.000,1000411172.500,0.9260,0
1000411242.500,1000411245.000,0.9089,0
```

Papers:

- **Detection of Einstein Telescope gravitational wave signals from binary black holes using deep learning,** Authors: Wathela Alhassan, T Bulik, M Suchenek
24/December/2022 MNRAS, stac3797, DOI: 10.1093/mnras/stac3797
- **PyMerger: Detecting Binary Black Hole merger from Einstein Telescope Using Deep Learning,** Authors: Wathela Alhassan, T Bulik, M Suchenek. ApJ, DOI: 10.3847/1538-4357/ad901e

Thank you!