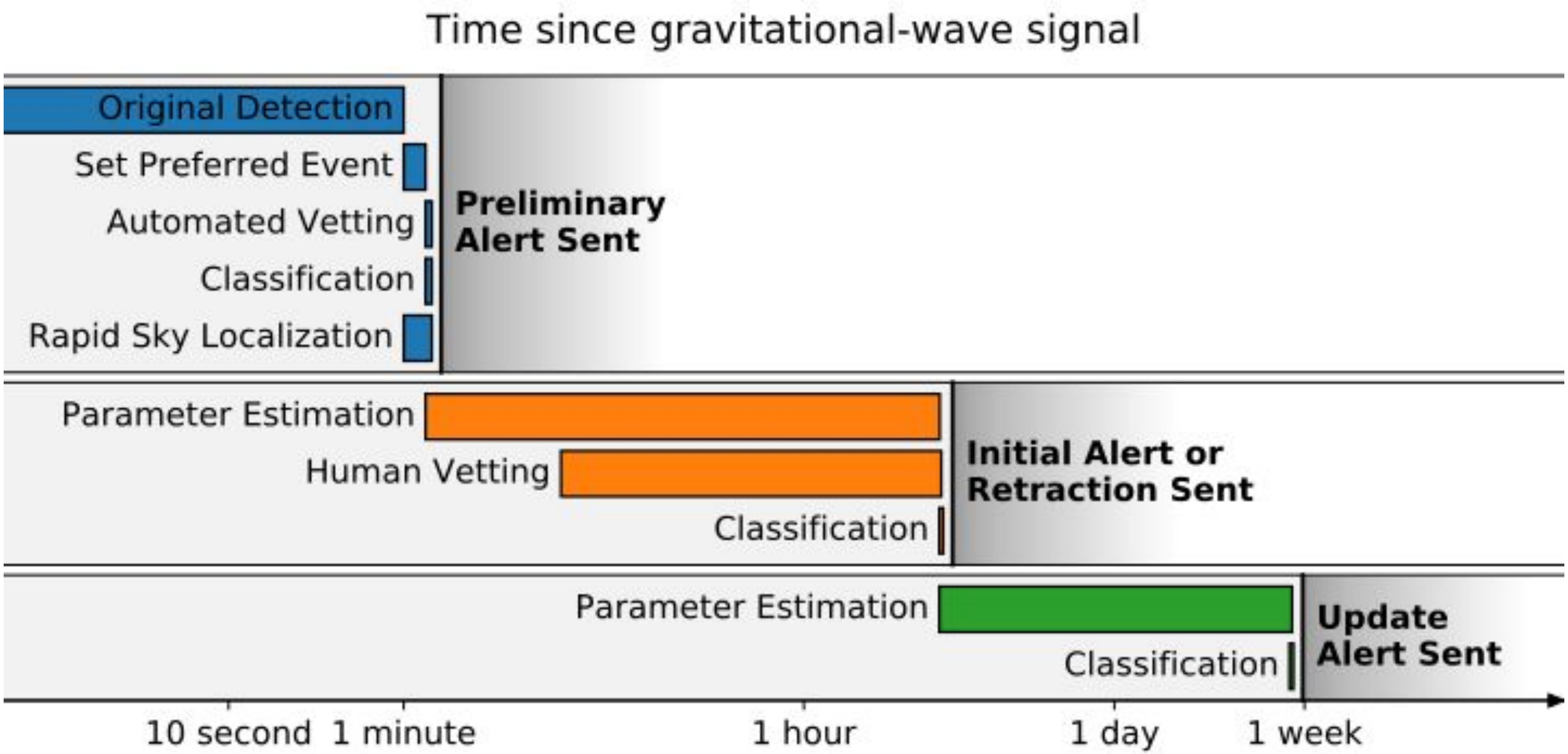
Low-latency Noise Mitigation Techniques in Gravitational-wave Detector Data Using Auxiliary Sensor Information

Patrick Godwin

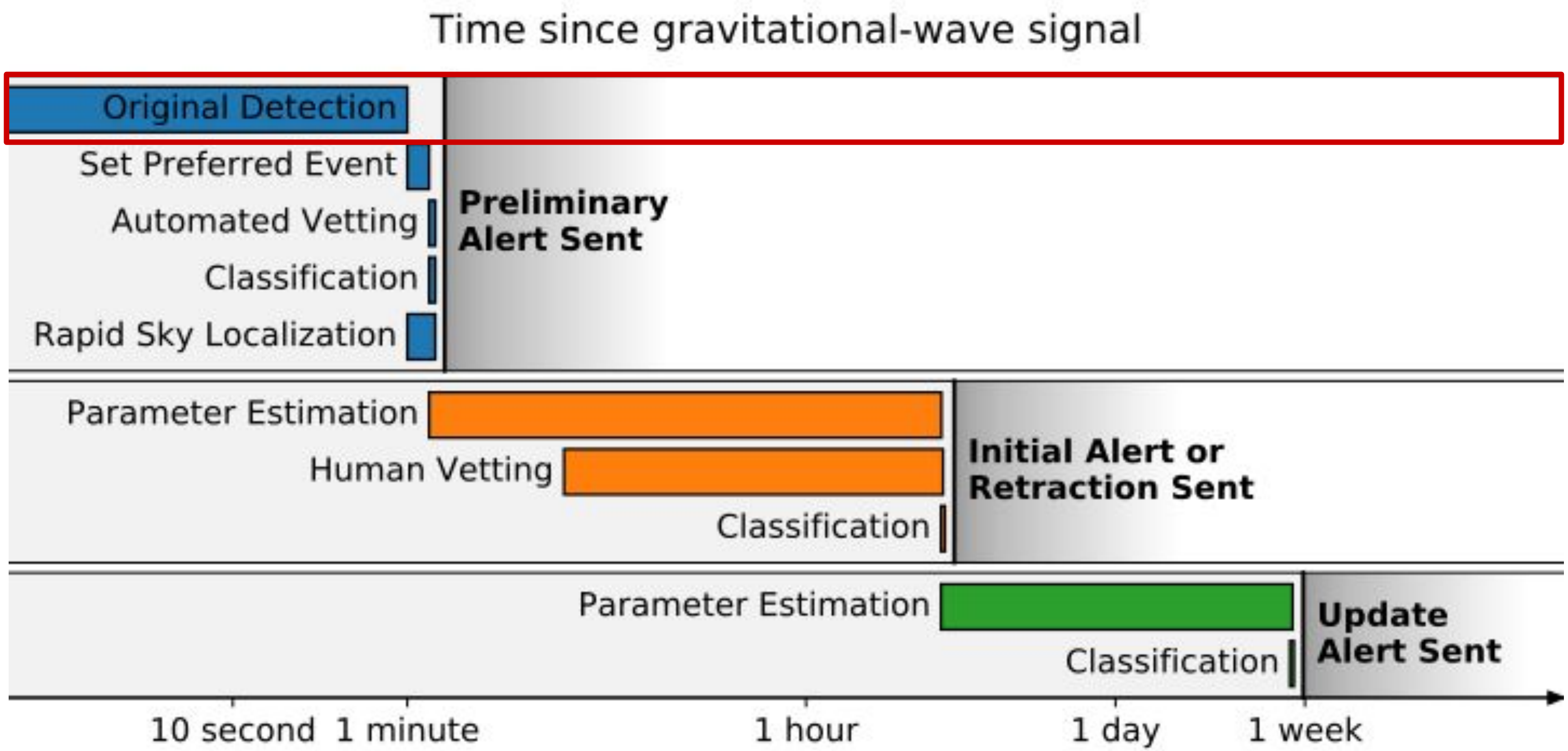
Department of Physics, Pennsylvania State University

Open Public Alert Timeline



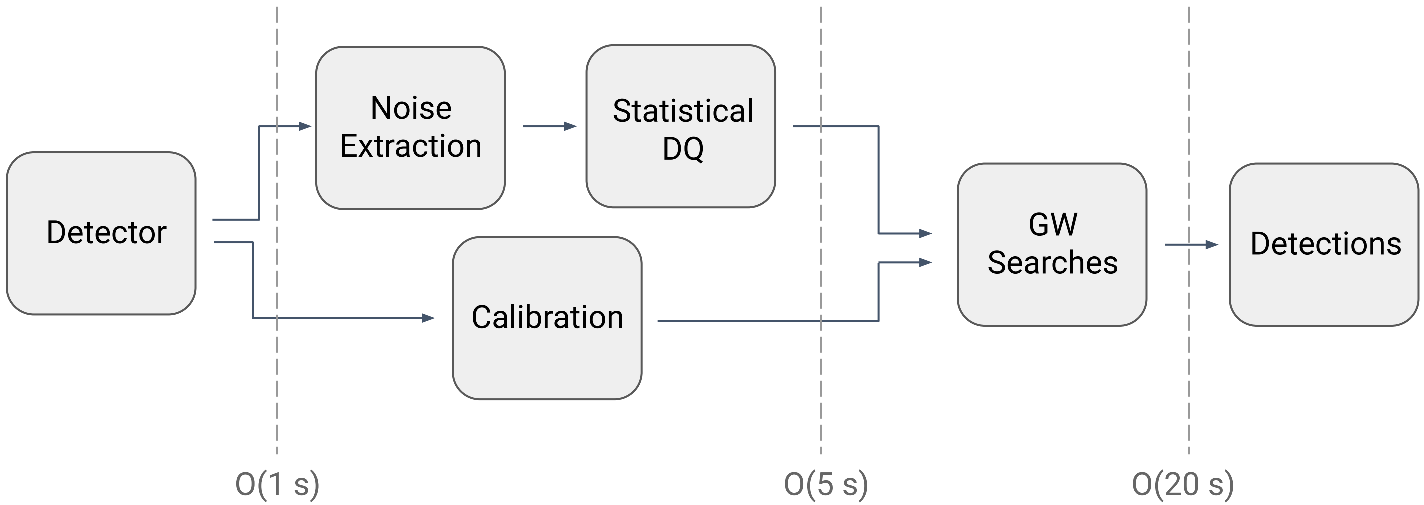
LSC Public Alerts User Guide, Version 12 (2019)

Open Public Alert Timeline

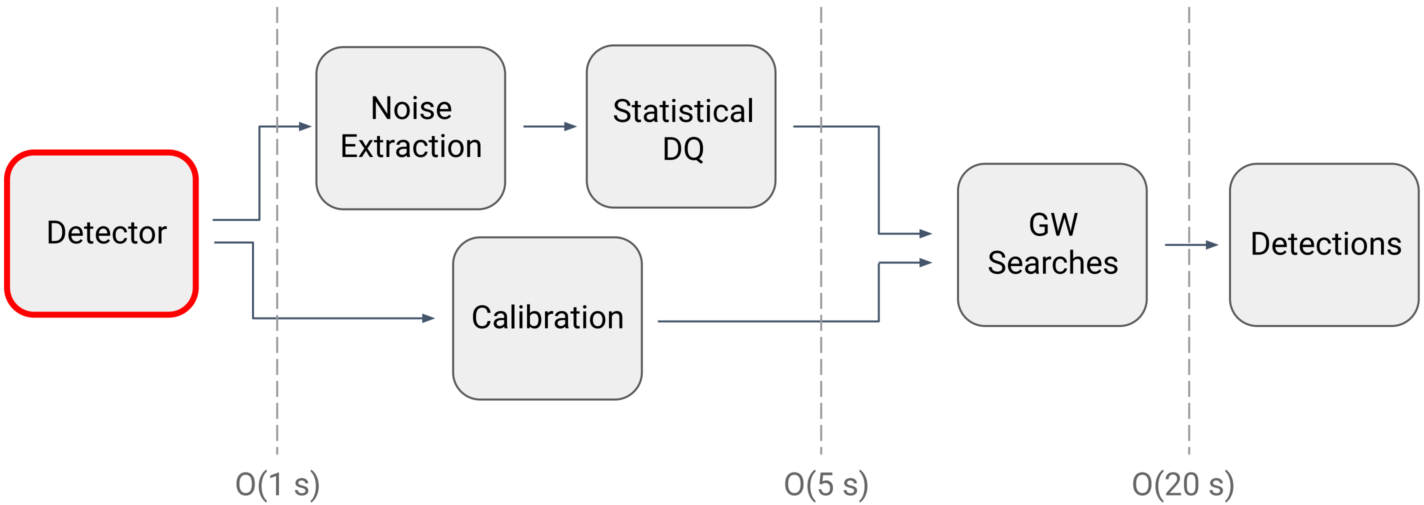


LSC Public Alerts User Guide, Version 12 (2019)

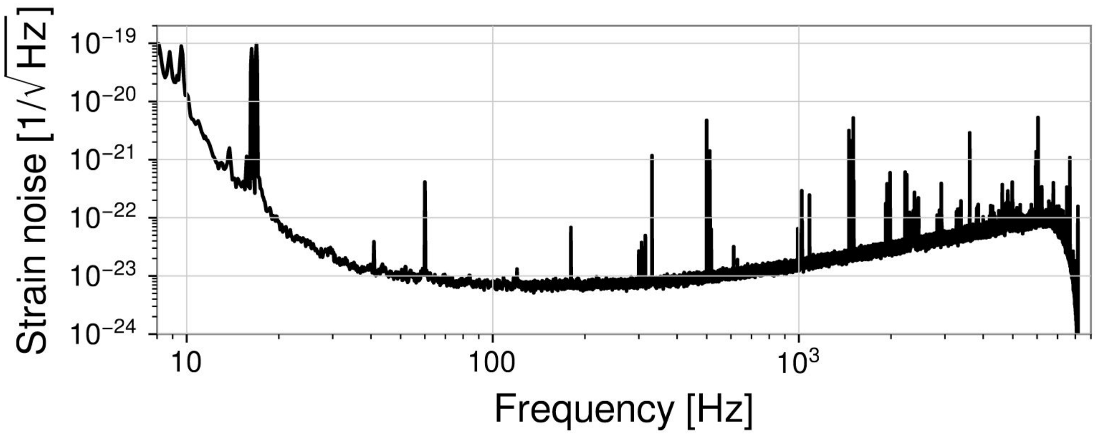
Simplified Low-Latency Dataflow: O3

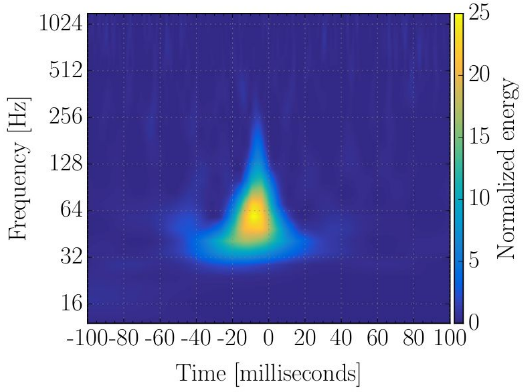


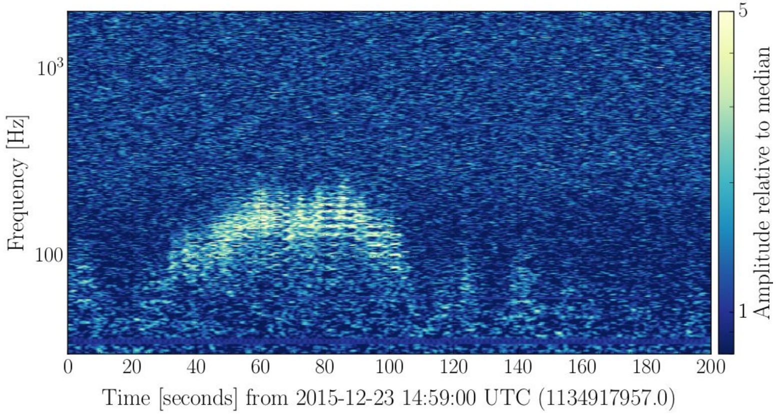
Detector Noise



Noise Types

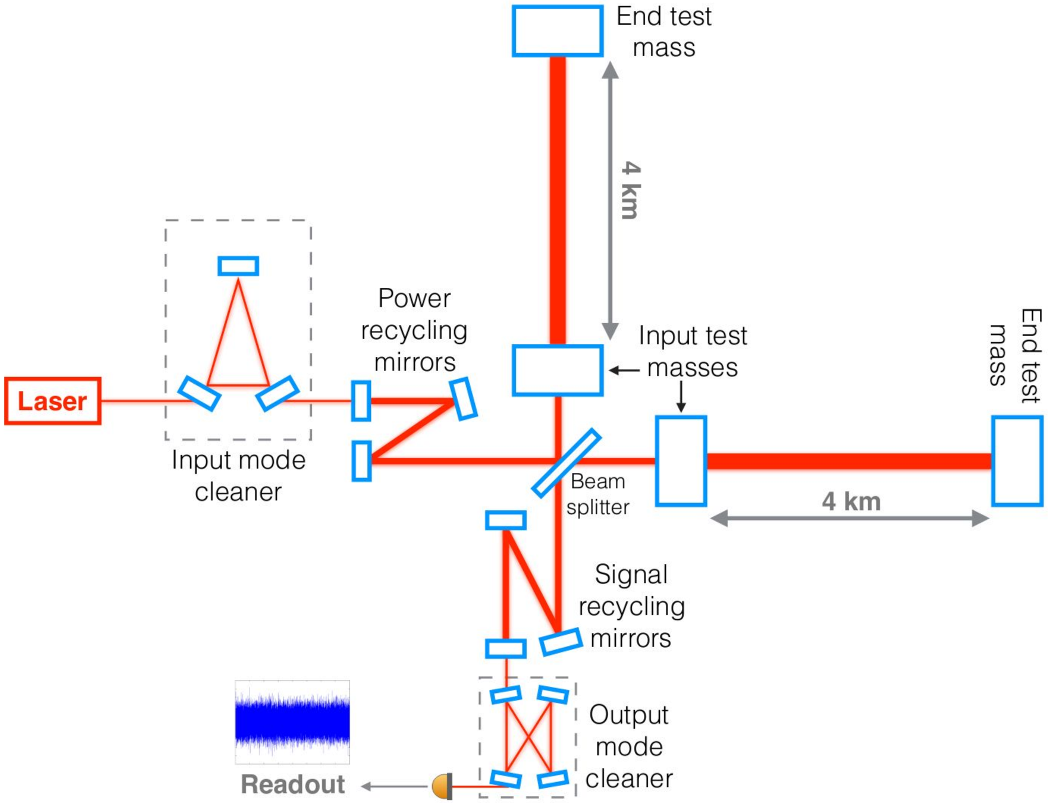




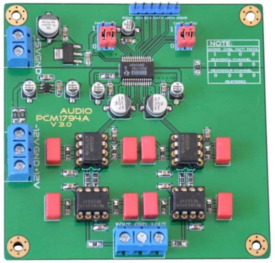


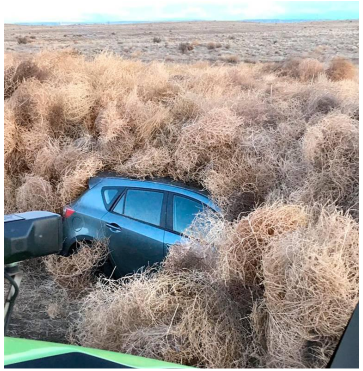
arXiv:1908.11170v2 + arXiv:1710.02185v4

Noise Sources

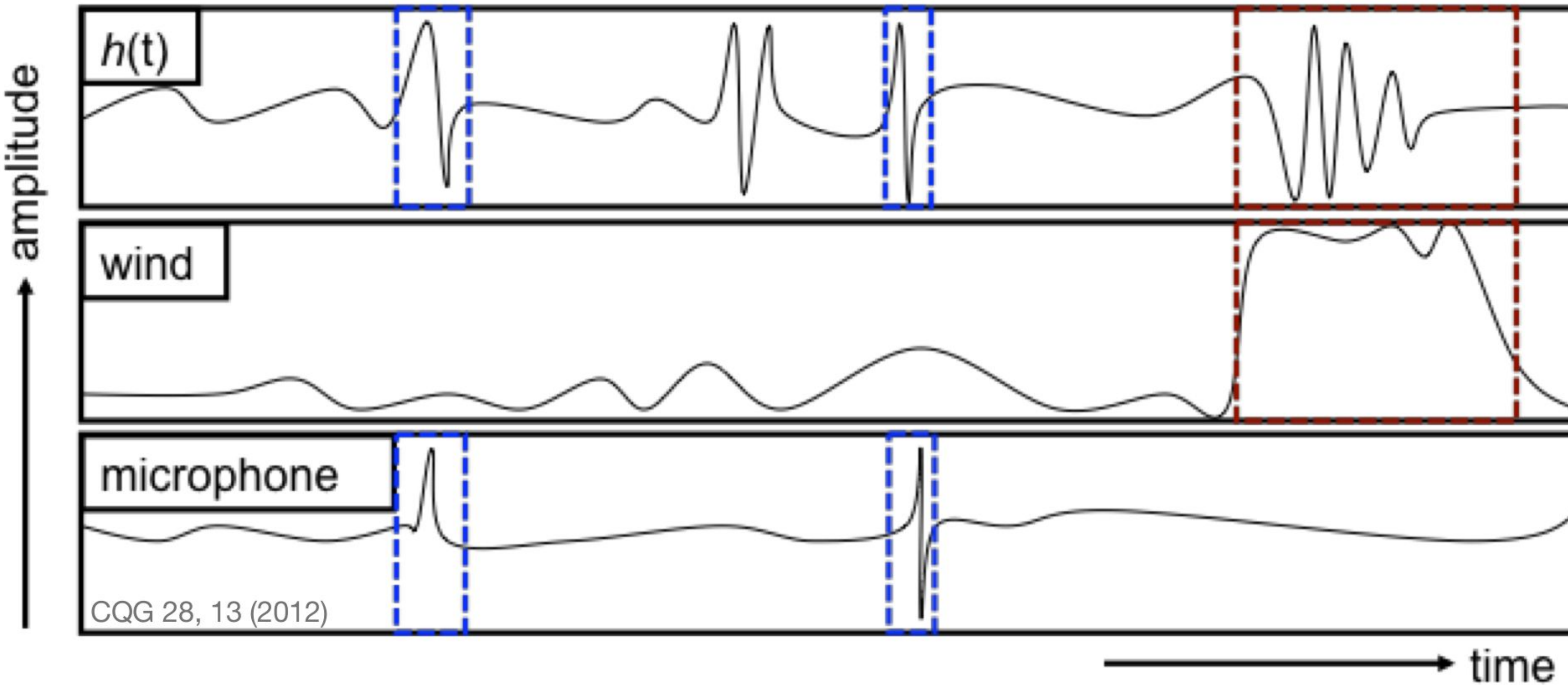


arXiv:1901.05093v2





Auxiliary Sensors



Auxiliary Sensors: A Big Data Challenge

- ~250k auxiliary channels
  - ~5k “fast” channels (≥ 256 Hz timeseries)
  - Rest are “slow” channels (16 Hz timeseries)
- Data rate: ~50 MB/s
- Data storage: ~ 1.5 PB/yr

Noise Mitigation Techniques

In order of preference:

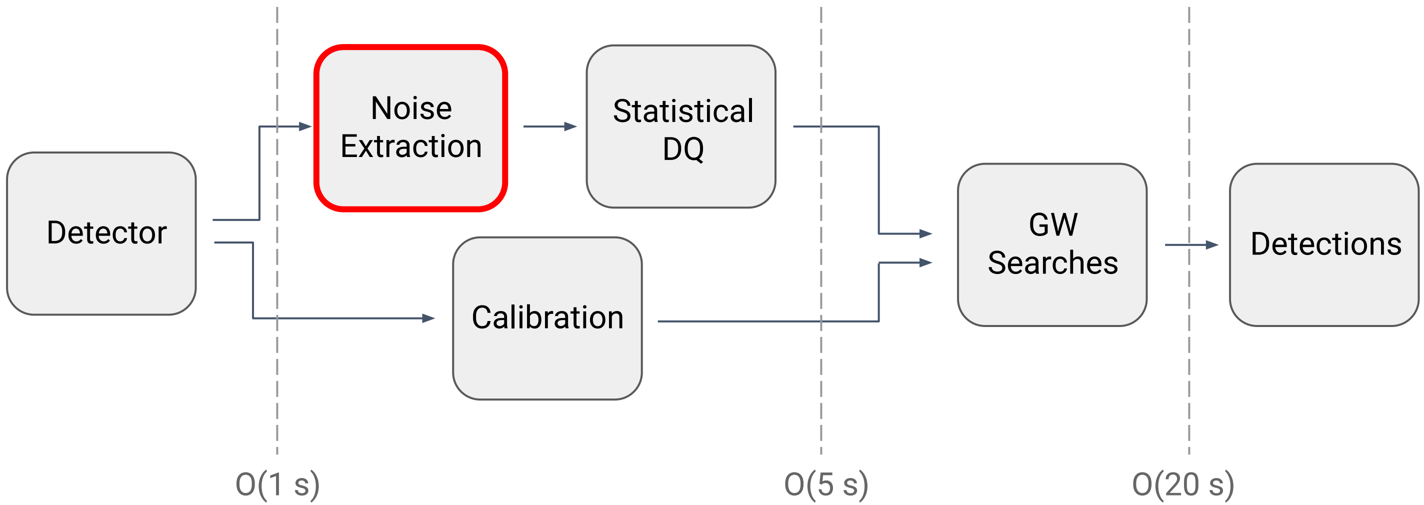
1. Noise Source Removal
2. Noise Subtraction
3. Noise Identification
4. Search-Specific Mitigation

Noise Mitigation Techniques

In order of preference:

1. Noise Source Removal
2. Noise Subtraction
3. Noise Identification
4. Search-Specific Mitigation

SNAX Toolkit



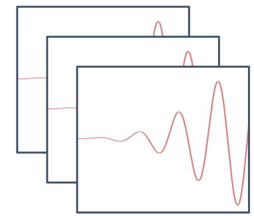
SNAX Toolkit: Overview

- Stream-based Noise Acquisition and eXtraction
- Extract non-stationary noise features in low-latency
- Sine-Gaussian basis
- **Output**: Multivariate timeseries data containing non-stationary noise features:

$$\vec{a}(t) = (\vec{a}_1(t), \ldots, \vec{a}_n(t))$$

feature vector

Auxiliary channel timeseries



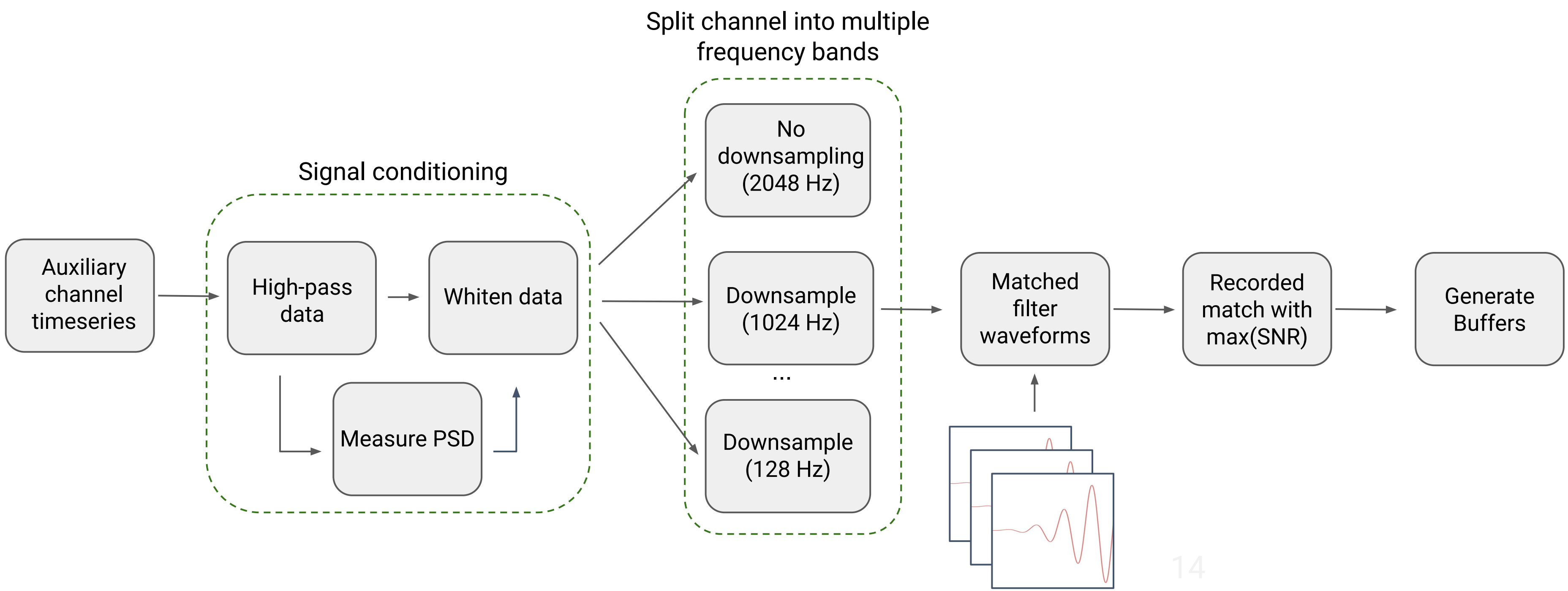
Matched filter waveforms

Record match with max(SNR) → $\vec{a}_i(t)$

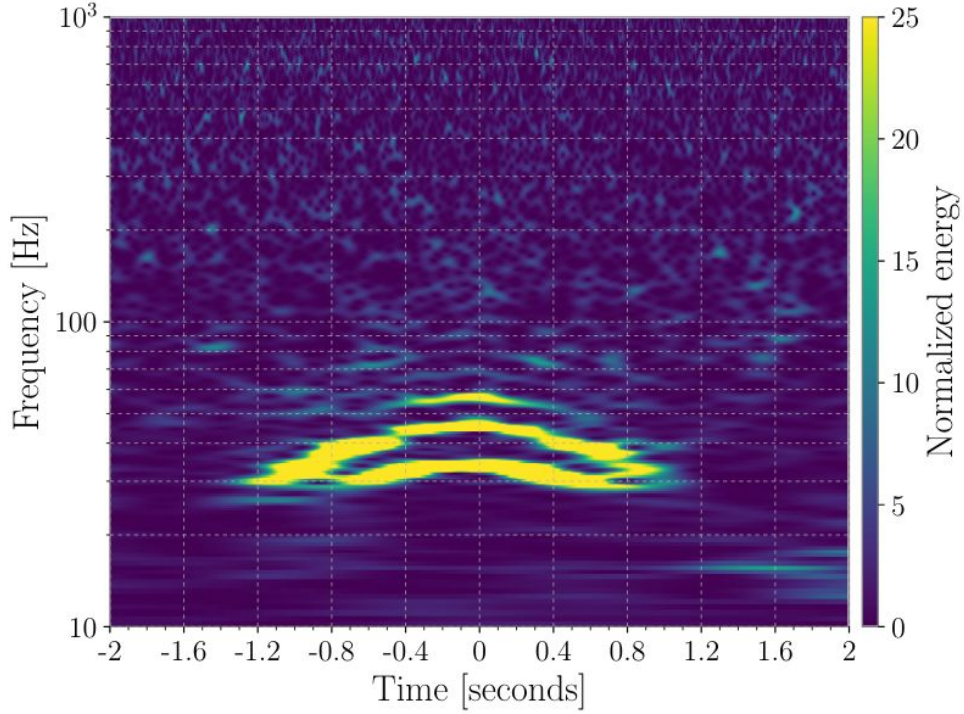
$$\vec{a}_i(t) = (\rho, \phi, t_{\mathrm{match}}, f, Q)$$

features

SNAX Toolkit: Workflow

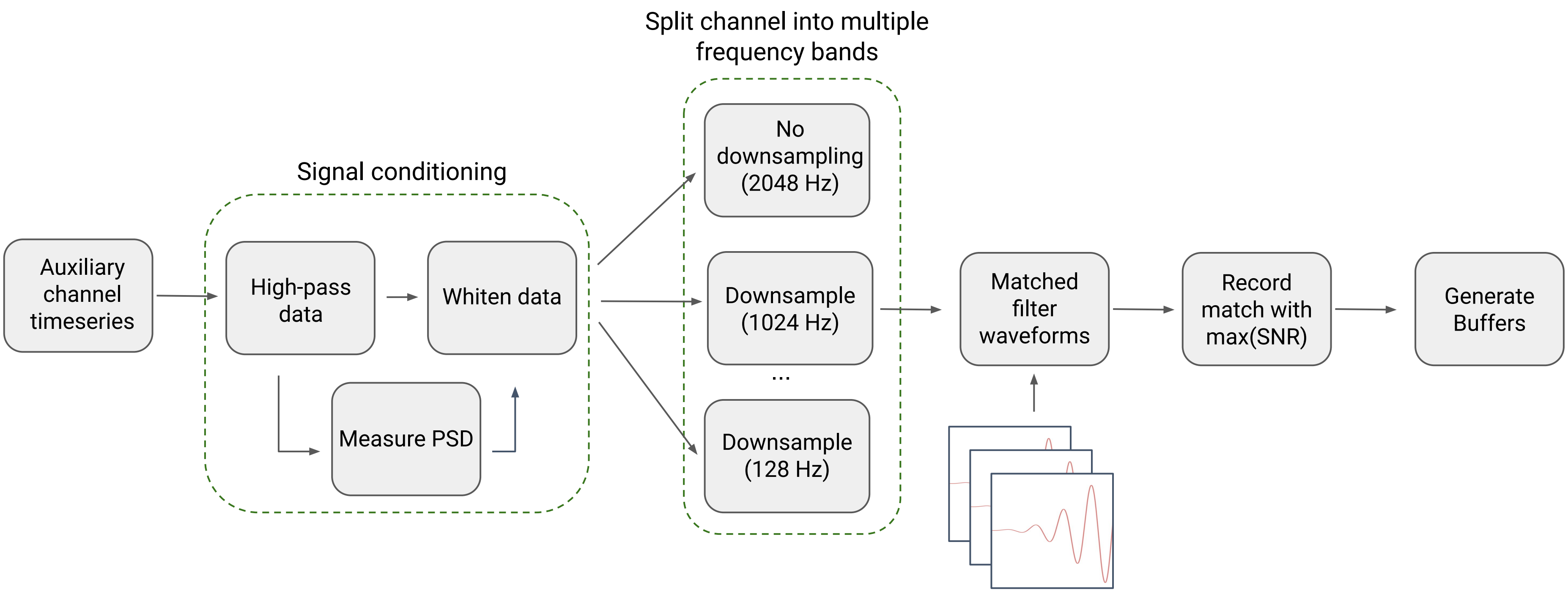


About Latency

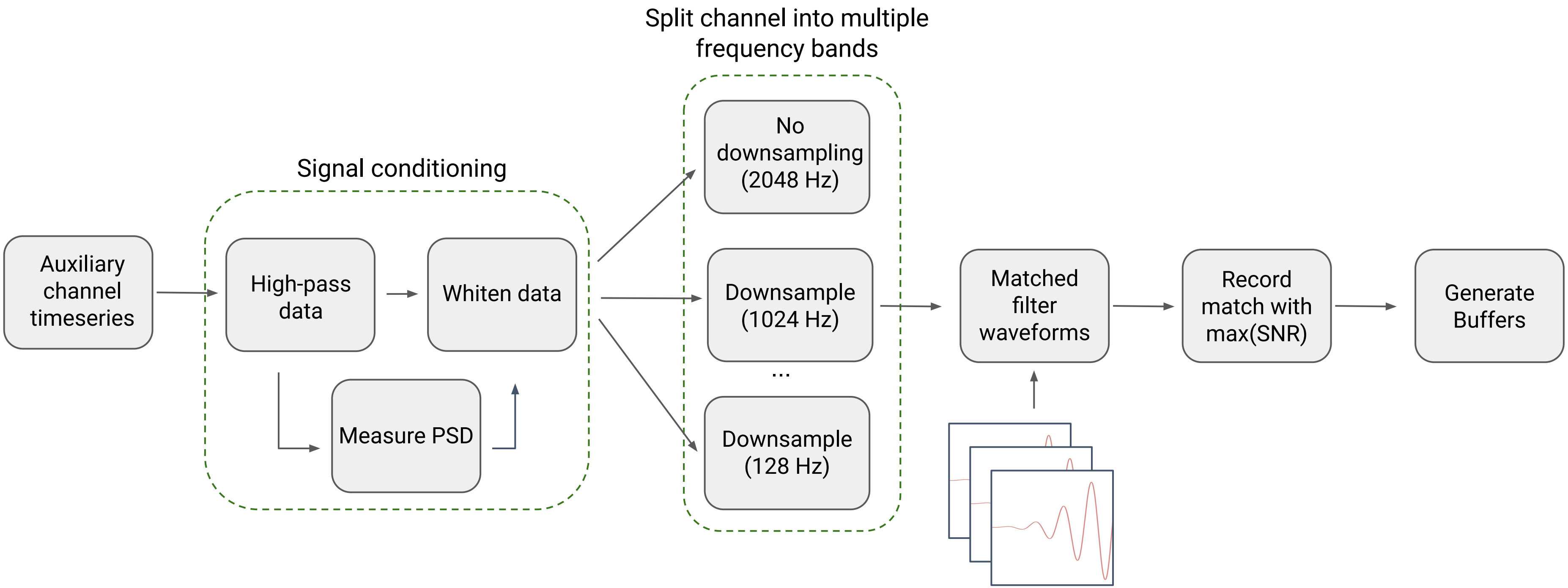


arXiv:1303.7159v1

SNAX Toolkit: Latency Budget

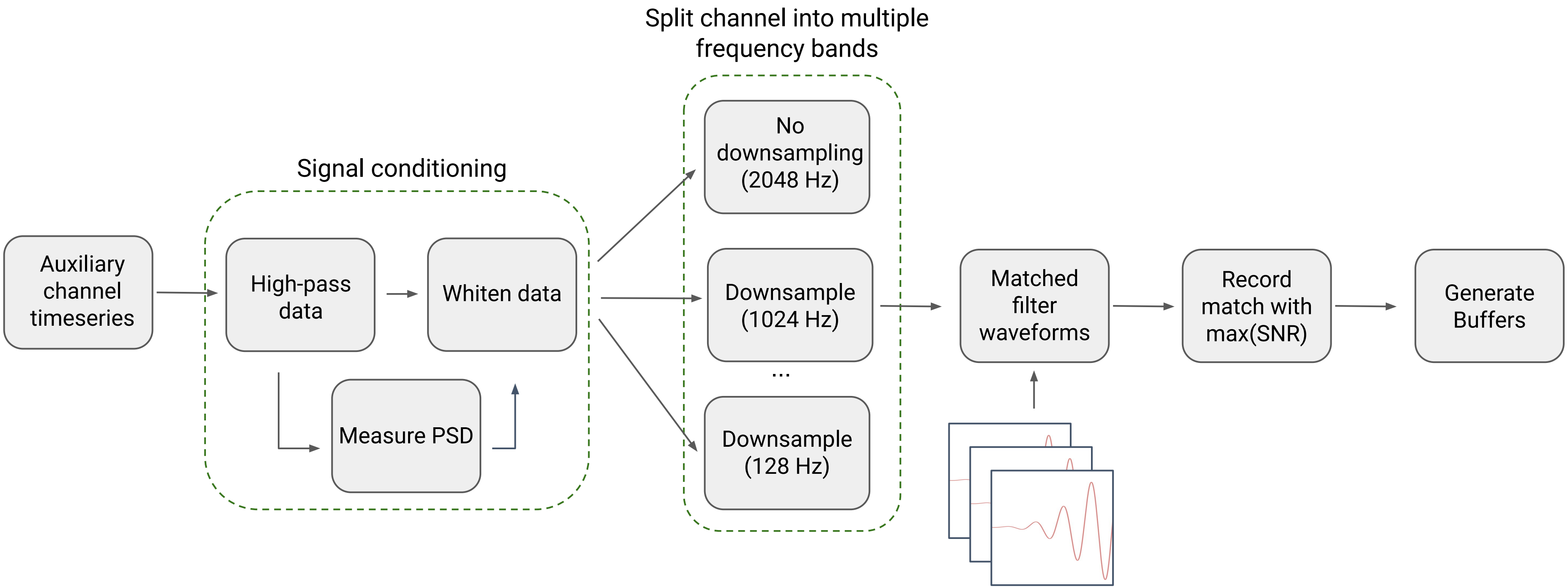


SNAX Toolkit: Latency Budget



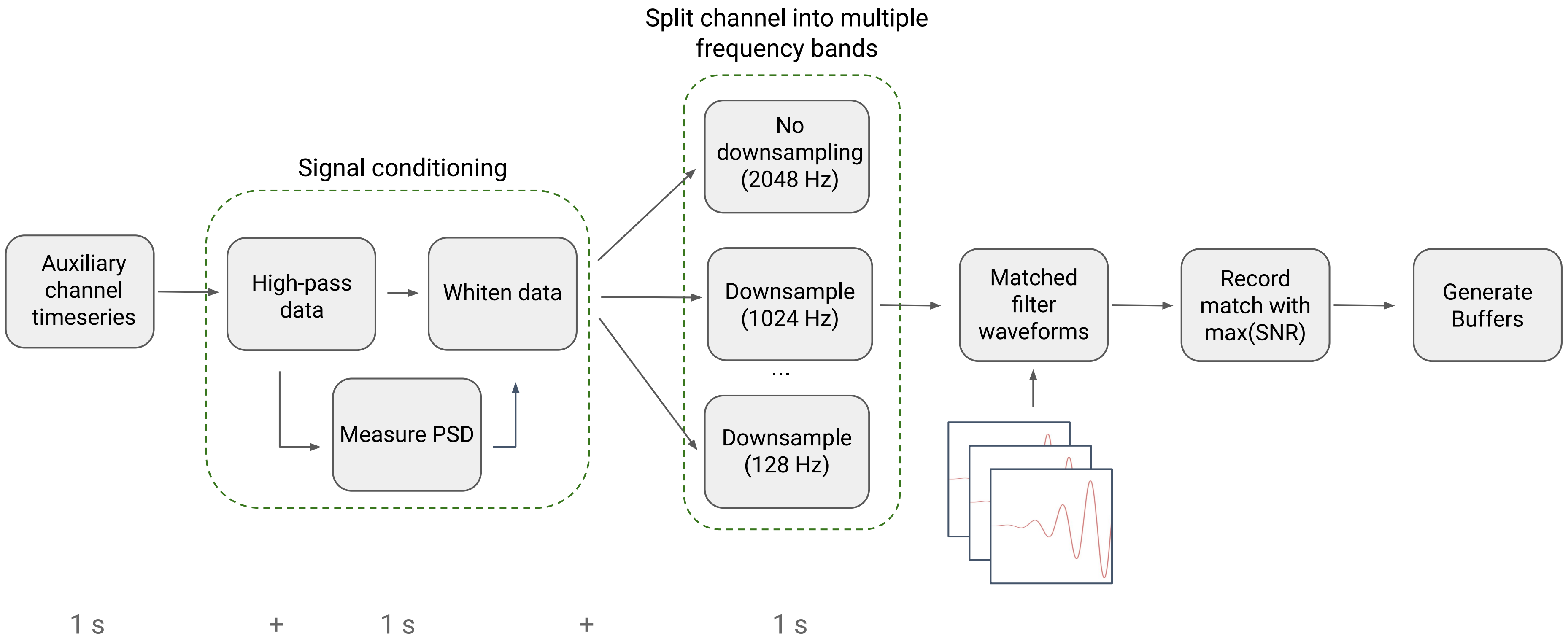
1 s

SNAX Toolkit: Latency Budget

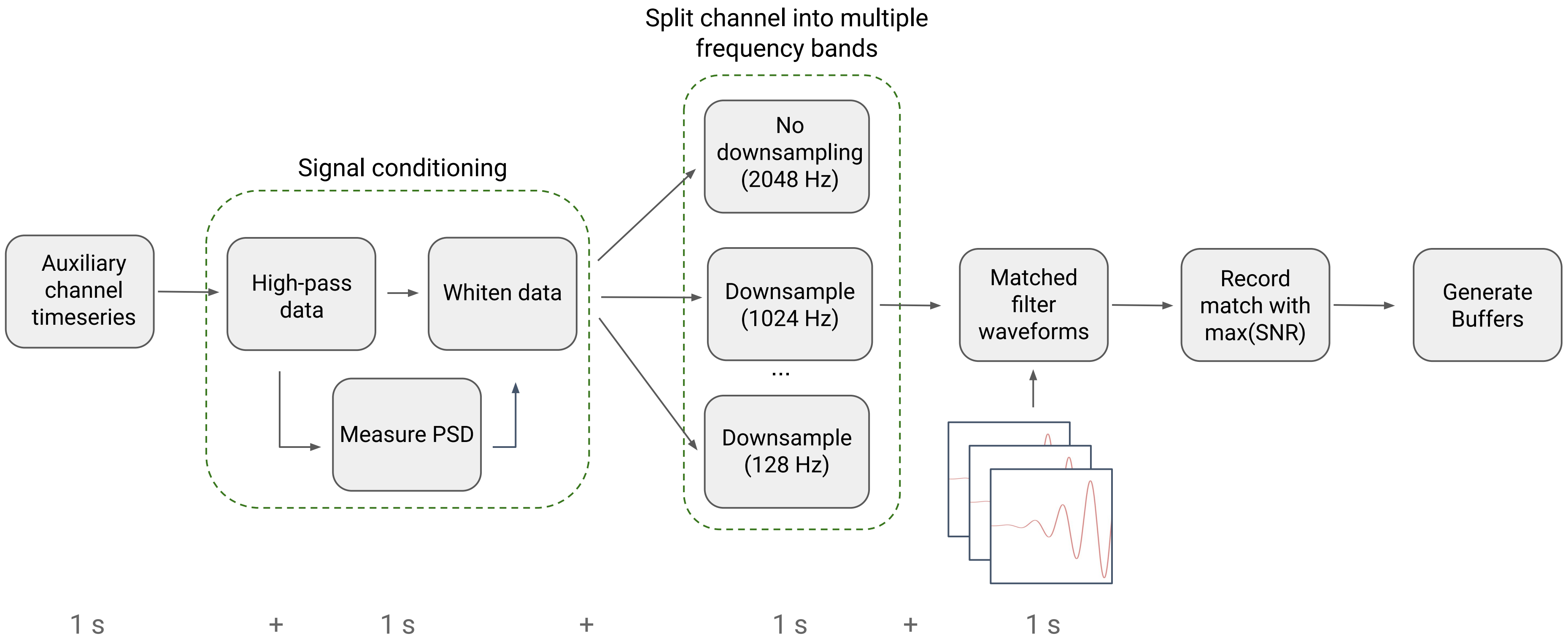


1 s + 1 s

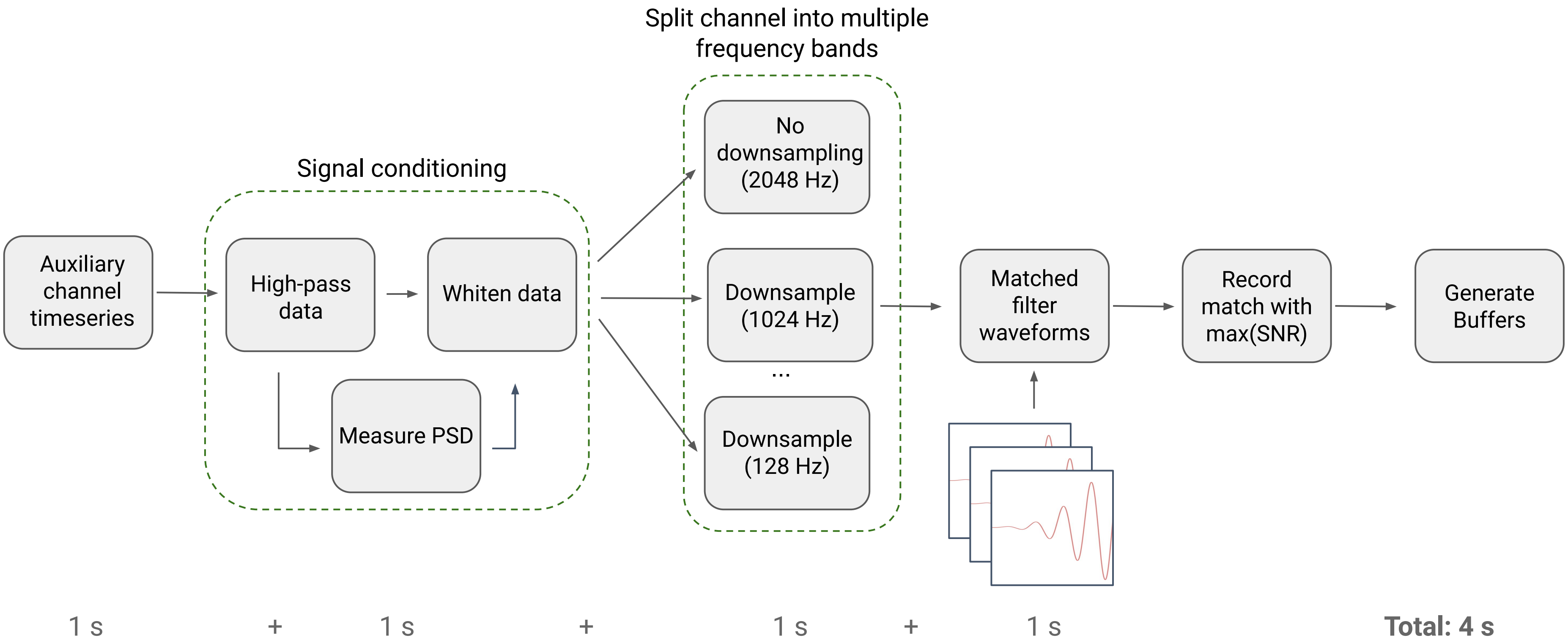
SNAX Toolkit: Latency Budget



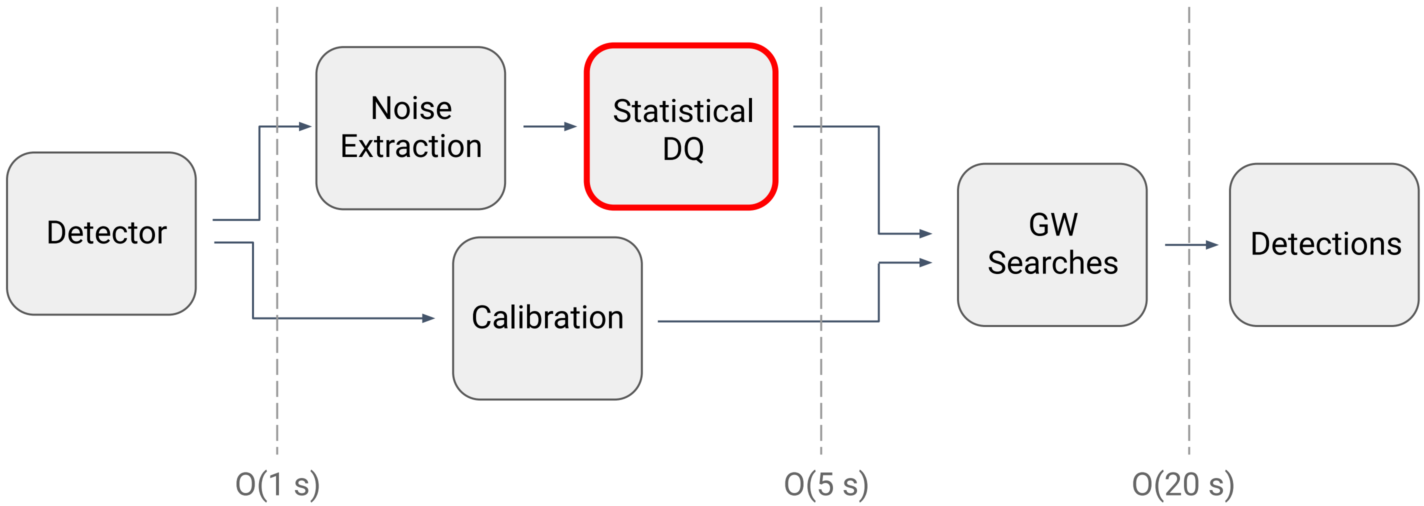
SNAX Toolkit: Latency Budget



SNAX Toolkit: Latency Budget



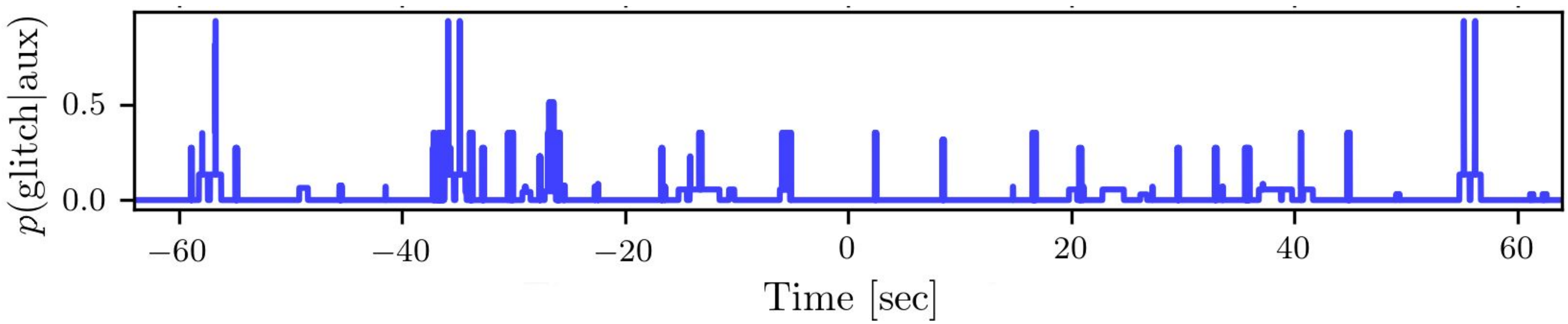
iDQ


Detector
Noise Extraction
Statistical DQ
Calibration
GW Searches
Detections
O(1 s)
O(5 s)
O(20 s)

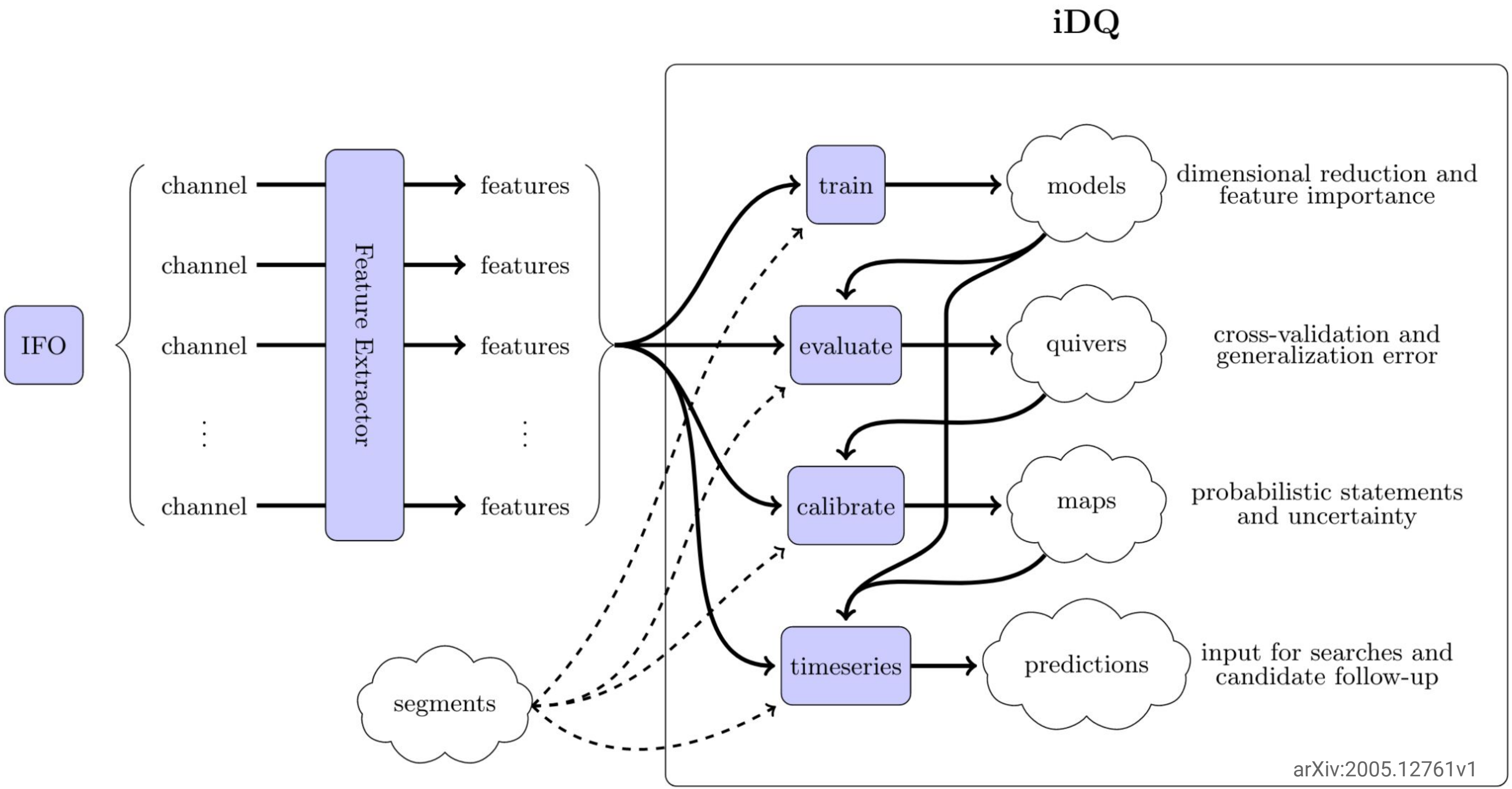
iDQ: Overview

- Search for correlations between auxiliary channels and *h(t)*
- **Output**: Probability that there is a glitch in *h(t)* as a function of time:

$$p_G(t) = p(G|\vec{a}(t)) = \frac{p(\vec{a}|G)p(G)}{p(\vec{a}|G)p(G) + p(\vec{a}|C)p(C)}$$



iDQ: Workflow



iDQ: Workflow

Features

$$\vec{a} \in \mathcal{R}^N$$

Models

$$\mathcal{M}_\alpha : \vec{a} \in \mathcal{R}^N \to r_\alpha \in [0, 1]$$

Maps

$$p(r_\alpha | G)$$

$$p(r_\alpha | C)$$

Predictions

$$\Lambda^G_C(r_\alpha) = \frac{p(r_\alpha|G)}{p(r_\alpha|C)}$$

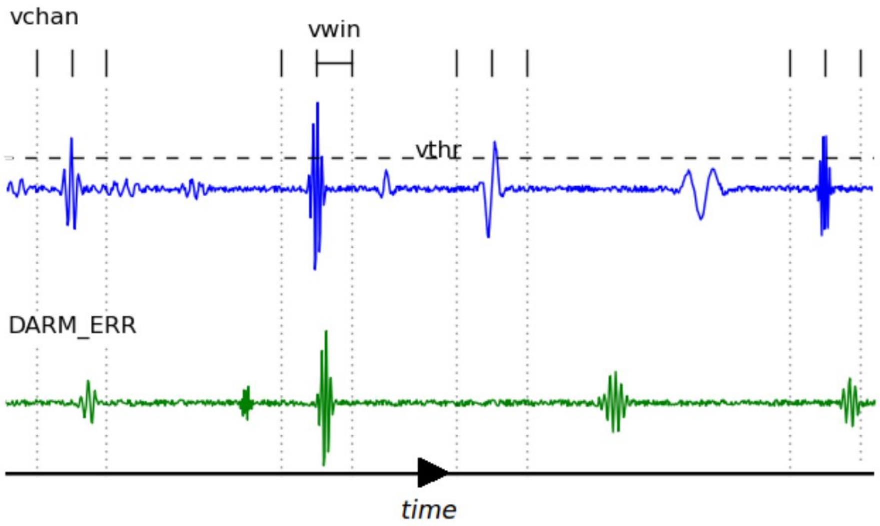
$$p_G = \frac{p(r_\alpha|G)p(G)}{p(r_\alpha|G)p(G) + p(r_\alpha|C)p(C)}$$

iDQ: Features

- Automatic retraining to deal with detector non-stationarity
- Provenance tracking
- Input features are relatively backend agnostic (SNAX, Omicron, etc)
- Supports variety of supervised classifiers
  - Statistical veto algorithms (OVL)
  - Classical ML classifiers (scikit-learn)
  - Gradient boosting (XGBoost)
  - Deep learning (Keras, PyTorch)

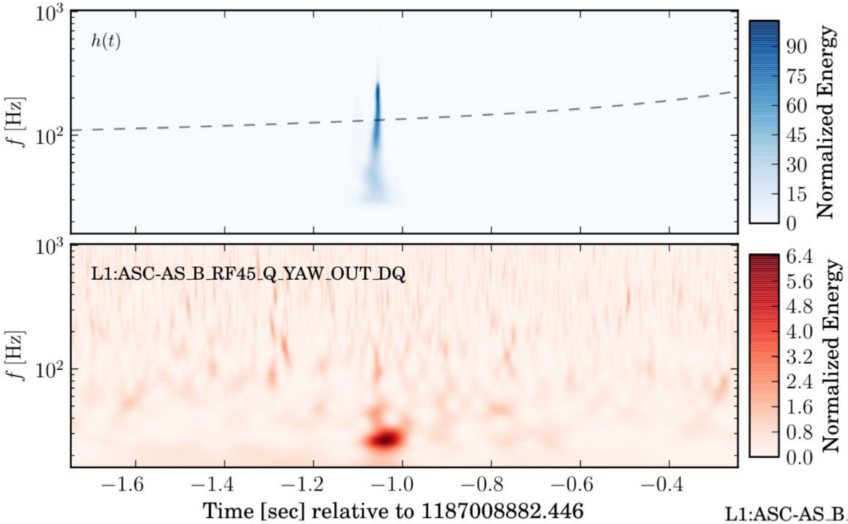
Ordered Veto List

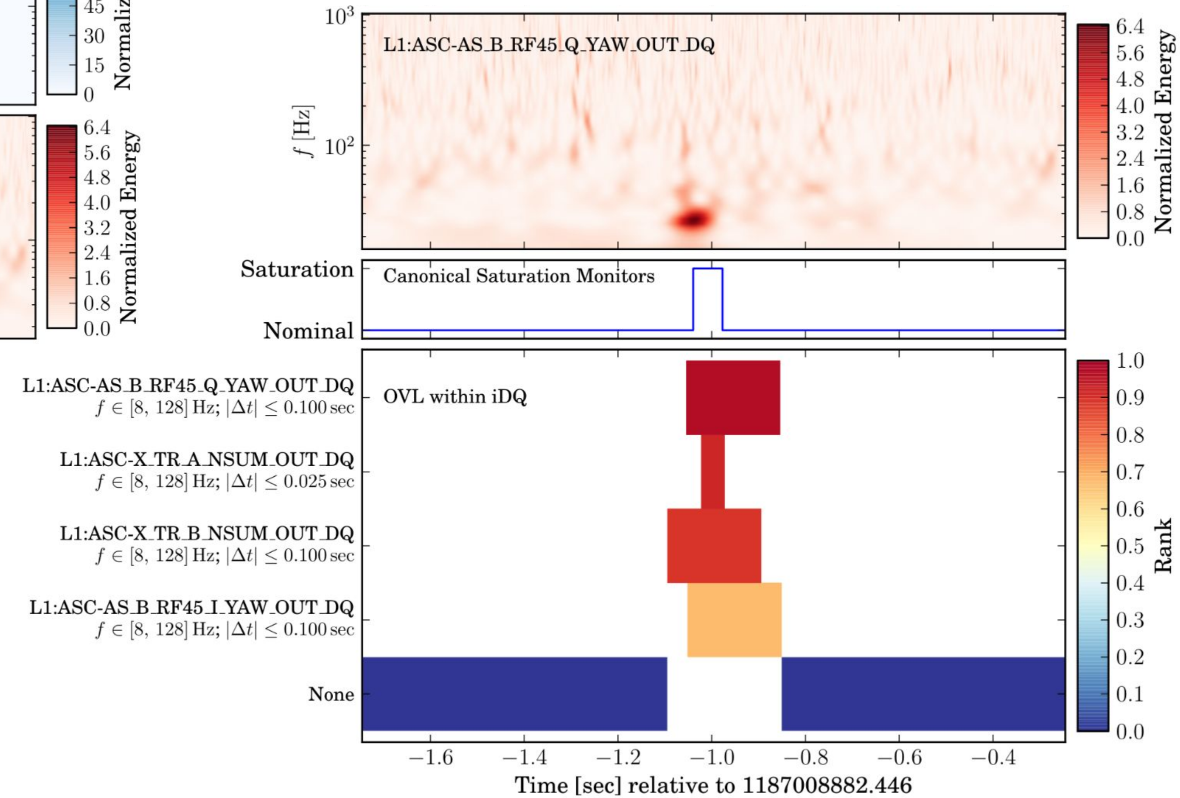
1. Apply veto configuration with a given threshold/window
2. Calculate veto efficiency for said configuration
3. Repeat for all threshold/window combinations for all auxiliary channels
4. Trim down set of veto configurations hierarchically



arXiv:1303.7159v2

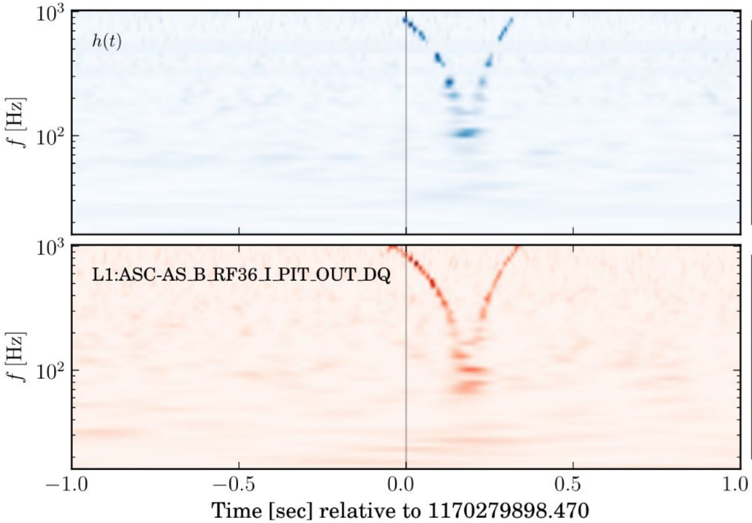
iDQ: GW170817

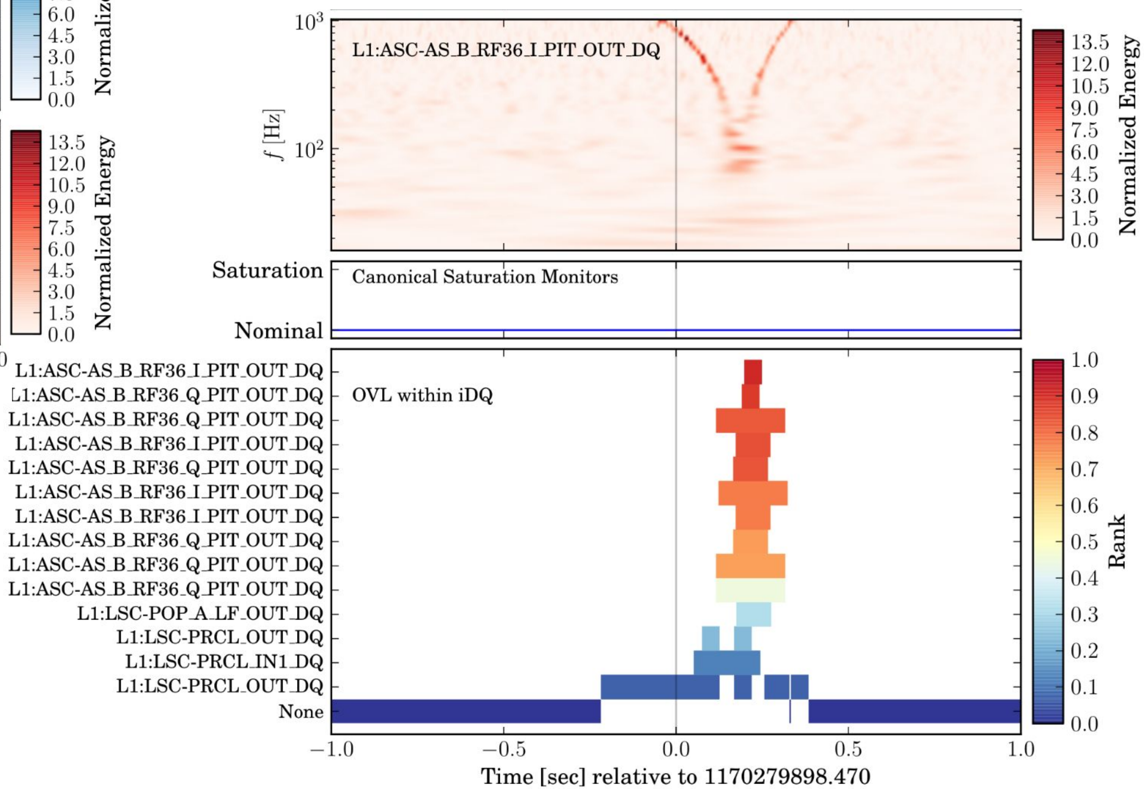




arXiv:2005.12761v1

iDQ: RF Whistle





arXiv:2005.12761v1

Wishlist For Online DQ

- Automated noise identification + mitigation (<5 second latency)
    - Medium-duration noise mitigation
    - Strain-only noise transient source mitigation
    - Noise subtraction for short duration transients
- Incorporation of non-binary DQ information into online searches
- Lock loss prediction