NONLINEAR REGRESSION WITH NEURAL NETWORKS

Rich Ormiston
Michael Coughlin
Rana Adhikari
Gabriele Vajente

August 20, 2018

PROJECT GOALS

- The primary objective of this project is:

Train neural networks to perform linear & nonlinear subtraction on time series data

- Relative to raw data, show that we can increase the detectable volume while simultaneously enhancing waveform recovery / parameter estimation

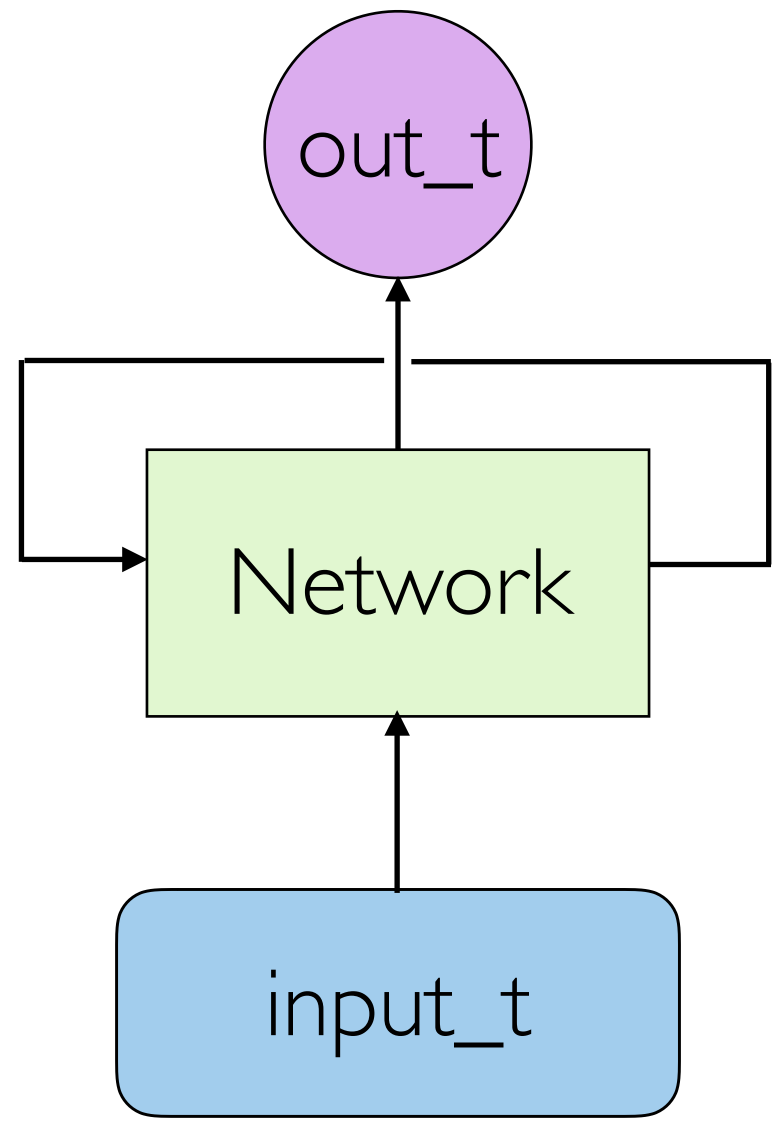
- In the limit of linear couplings, show that we reproduce the Wiener filter results

- Clean up the ~10-80 Hz frequency band

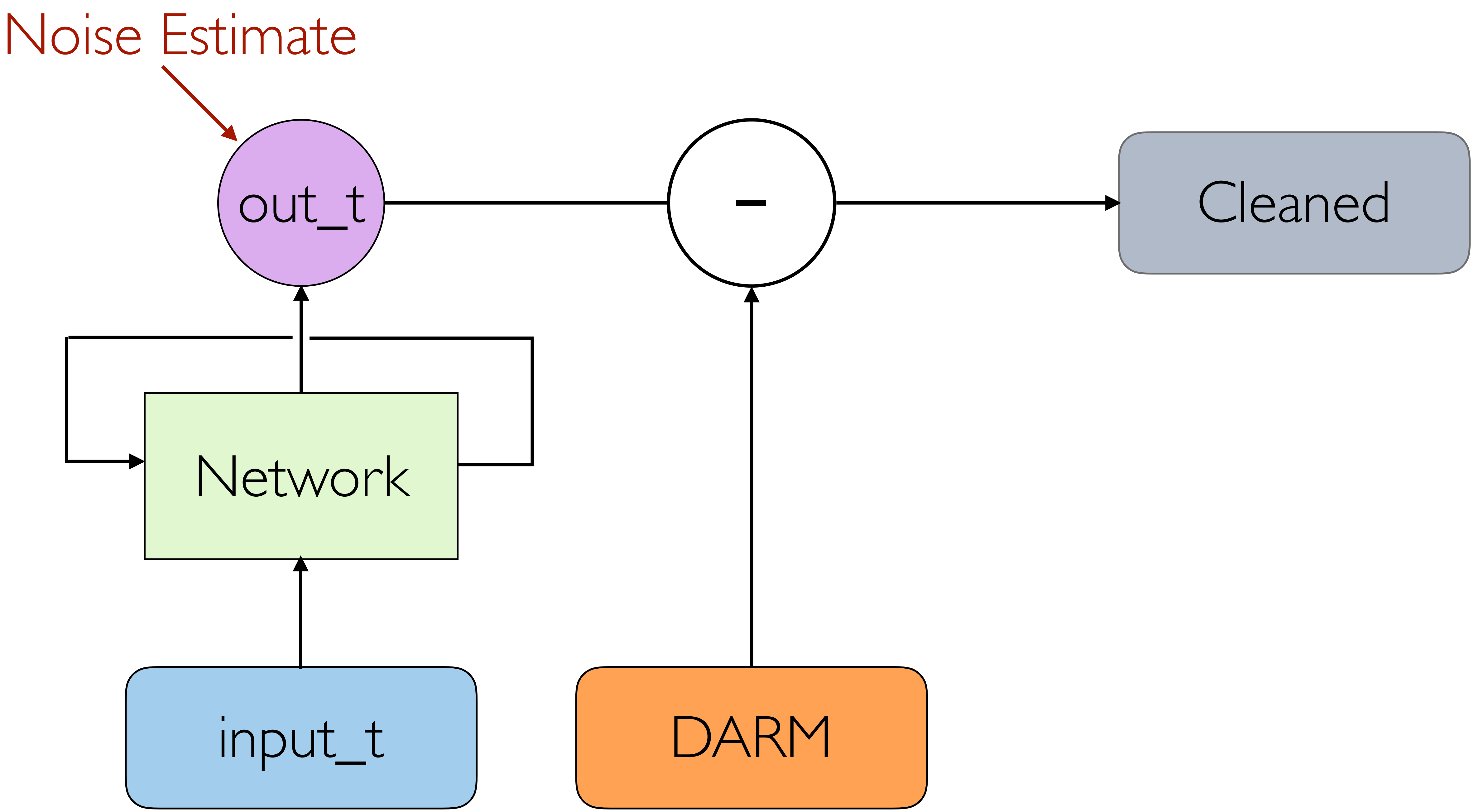
BRIEF OVERVIEW OF RECURRENT NEURAL NETWORKS

BASIC RNN

- Data from the input channels at time "t" are fed into the network.
- The network updates and then produces an output point estimate of the target out_t.
- The output is also fed back into the network so that the input at time step "t+1" *and* the previous output (out_t) are both used to inform the next output point estimate out_{t+1}



RNN FOR DATA QUALITY

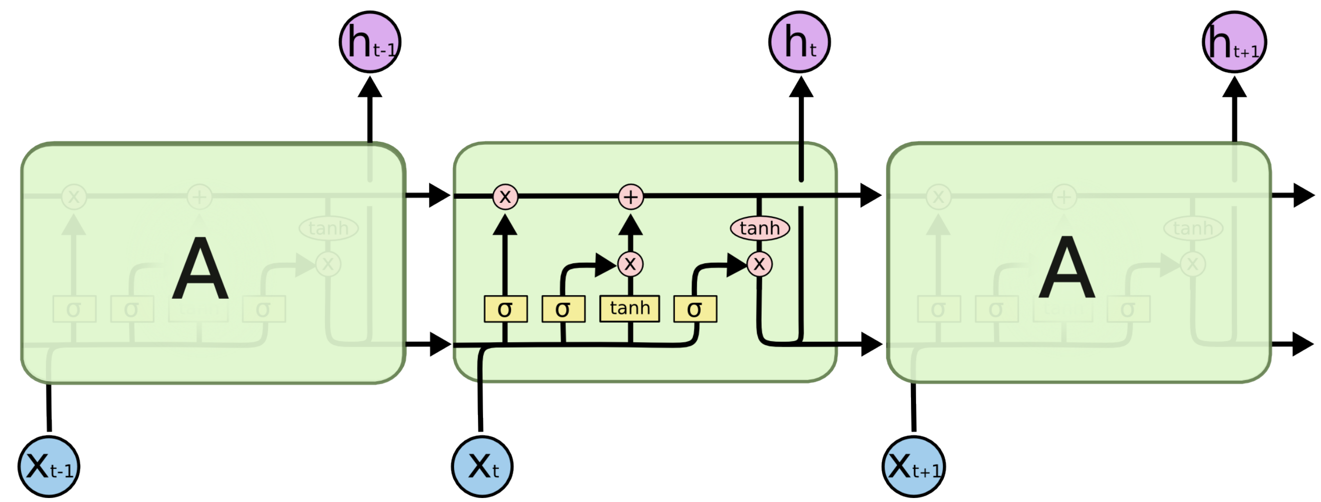


LSTM NETWORKS

Long Short-Term Memory Networks are a subset of RNNs

Each input time step is analyzed through a series of gates (forget, input, output) and this information is fed into a "cell state."

The output is then fed into the next input along with the next time step → the updated network "remembers" the past.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

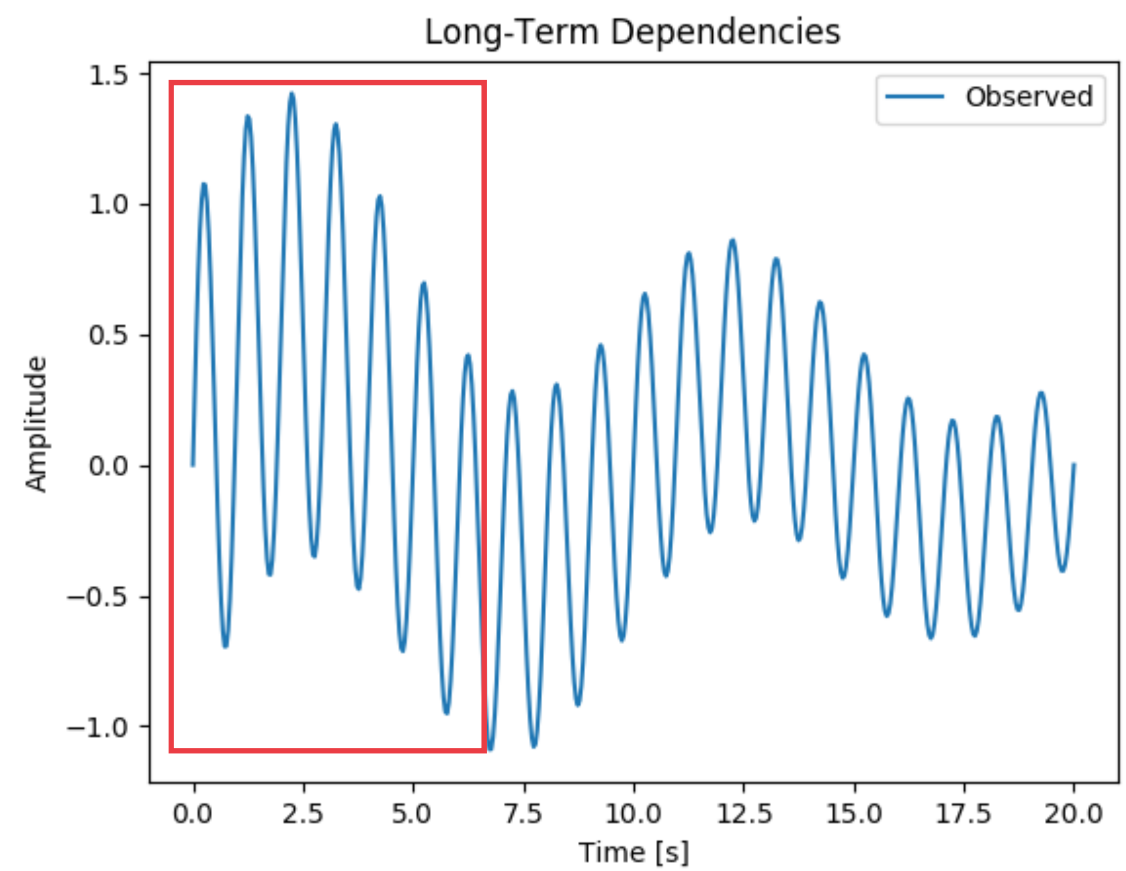
$$h_t = o_t \cdot \tanh(C_t)$$

LSTM Flow Control (See Chris Olah's Blog)

CAPTURING LONG-TERM DEPENDENCIES

Question:
Would the NN be able to figure out the damping factor if it only saw what was in the red box?

Answer:
With an LSTM, it could. But almost certainly not with a standard feed-forward network of fully connected layers



$$\text{Observed} = e^{-0.1t}\left(\frac{3}{5}\sin(10t) + \sin(t)\right)$$

WHY DO WE NEED LSTMs?

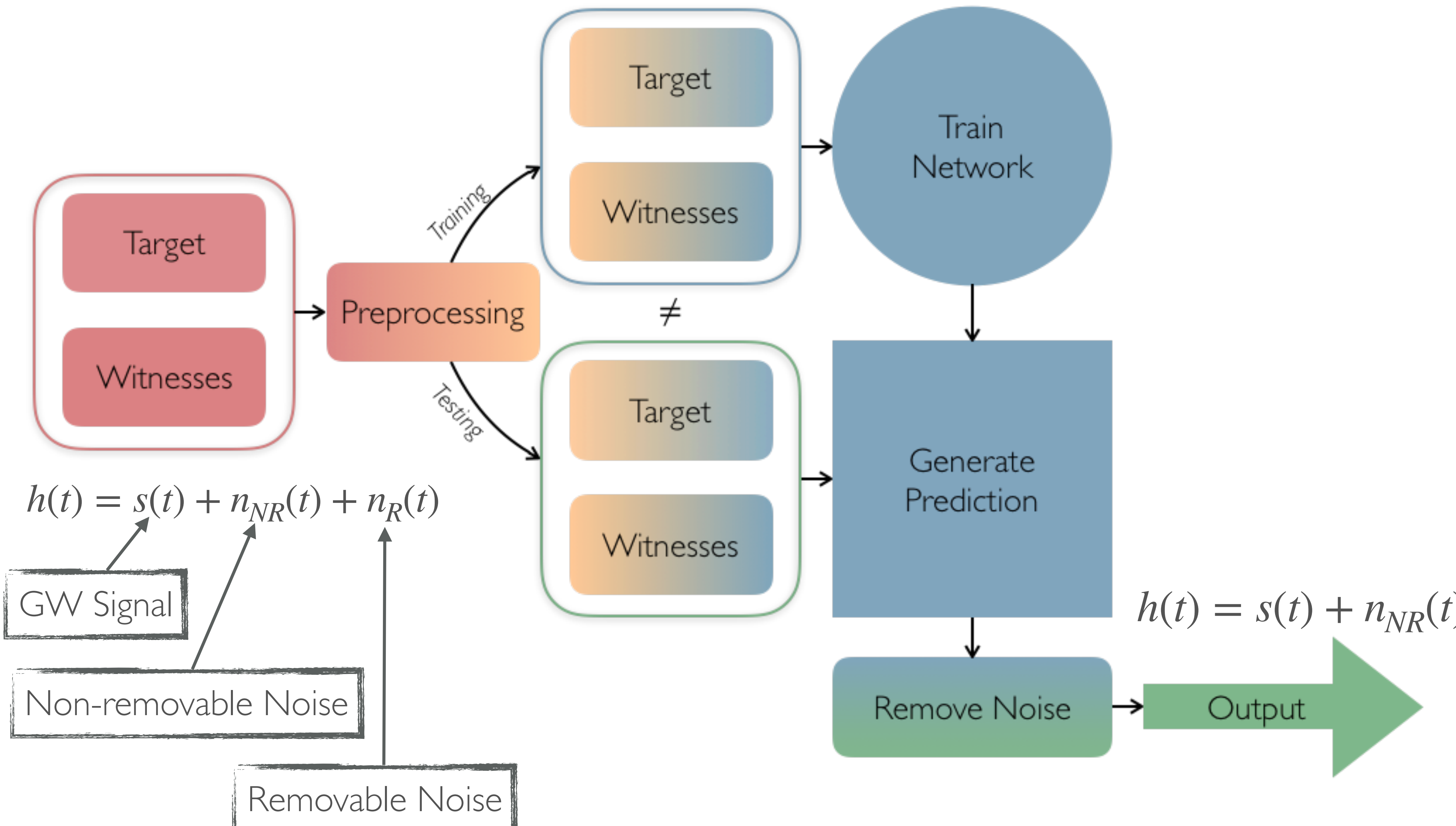
- Seismic waves at the test masses will take several seconds to get to the corner station.
- Slowly migrating signals (e.g. wandering lines).

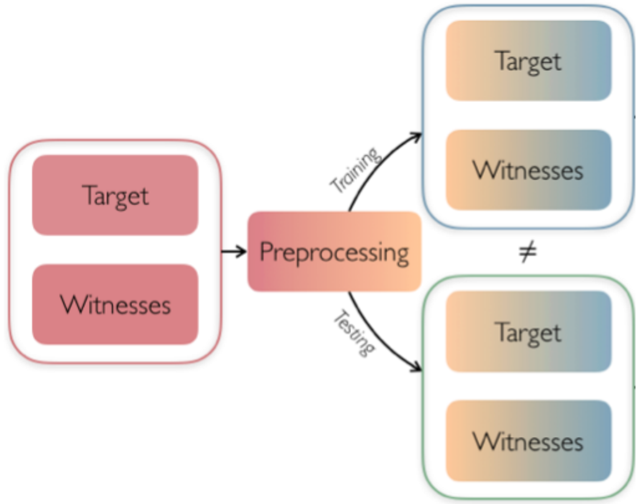
THE DEEP CLEAN REPOSITORY

DEEPCLEAN OVERVIEW

- Website, Repo and Docs
- Deep learning repo built on Tensorflow (pyTorch coming soon)
- Streams from nds2 or loads mat files
- Easy config file usage (no MLA knowledge prerequisite)
- Trains models, cleans data, and saves the cleaned output
- Automatically generates webpages
- Hyperparameter tuning scripts
- Data visualization and Spearman / Pearson tests

DEEPCLEAN WORKFLOW I





WORKFLOW II

- Separate out the target from the reference data.
- Preprocessing can involve many steps. Generally, we normalize (or standardize):

$$Data \rightarrow \frac{Data - \mu}{\sigma}$$

To reduce training time and increase performance, we also bandpass frequency bands and "loop" over the network performing subtraction in each band separately

- For supervised learning, split the data into training and testing samples. Network never "sees" testing data

WORKFLOW III

Network training consists (roughly) of three parts:

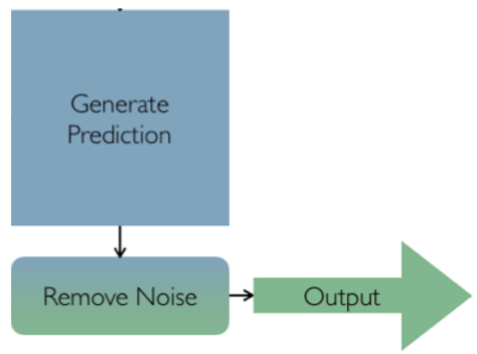
1. Given the witnesses and lookback L, $(\bar{\theta}_t, \ldots \bar{\theta}_{t-L})$, calculate a prediction $\tilde{D}_t$ (repeat for each time step)

$$NN[\bar{\theta}_t, \bar{\theta}_{t-1}, \ldots \bar{\theta}_{t-L}] = \tilde{D}_t$$

2. Calculate the error of the prediction with the target value through a cost function $C(D_t, \tilde{D}_t)$

3. Update the weights to minimize the cost function

$$\bar{w} \to \bar{w} - \eta \bar{\nabla} C(D_t, \tilde{D}_t)$$



WORKFLOW VI

After training, we can feed in the test data and generate a prediction.

For DeepClean, this point is a little subtle. We want to predict the "subtractable*" noise part of h(t), but this is not available as a target! We only have the full strain channel, h(t), giving us an approximate target. (see extra slides)

The prediction should be an estimate of the removable noise only. So target-prediction is the cleaned network output.

*Non-subtractable noise would be noises for which there is no witness or noises which are random (e.g., shot noise)

DEEP CLEAN

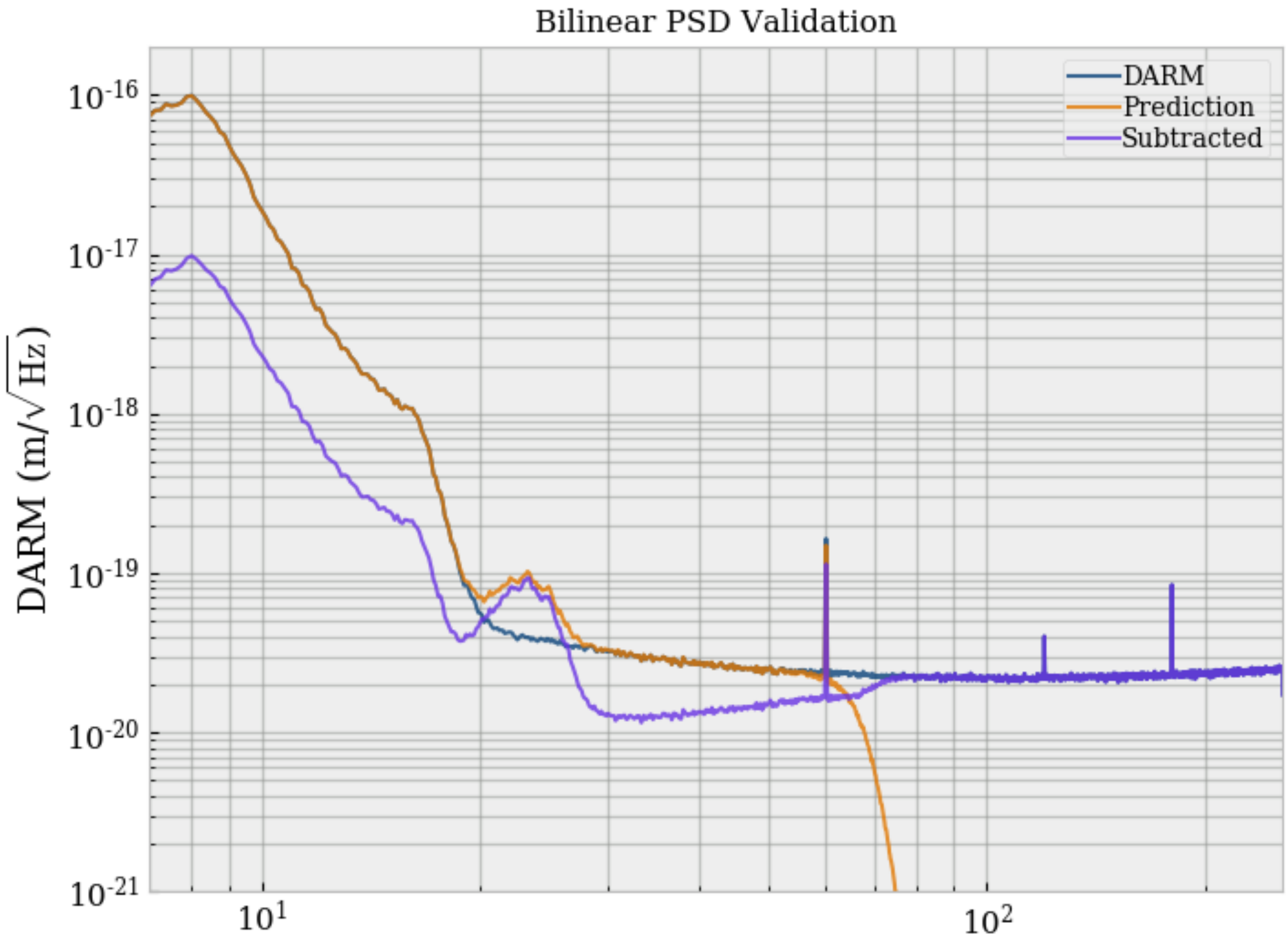
RESULTS

MOCK DATA TEST

Jitter Noise $= j_1(t), j_2(t)$

DARM $= h(t) + \alpha * j_1(t) * j_2(t)$

Witnesses $= [j_1(t), j_2(t)]$

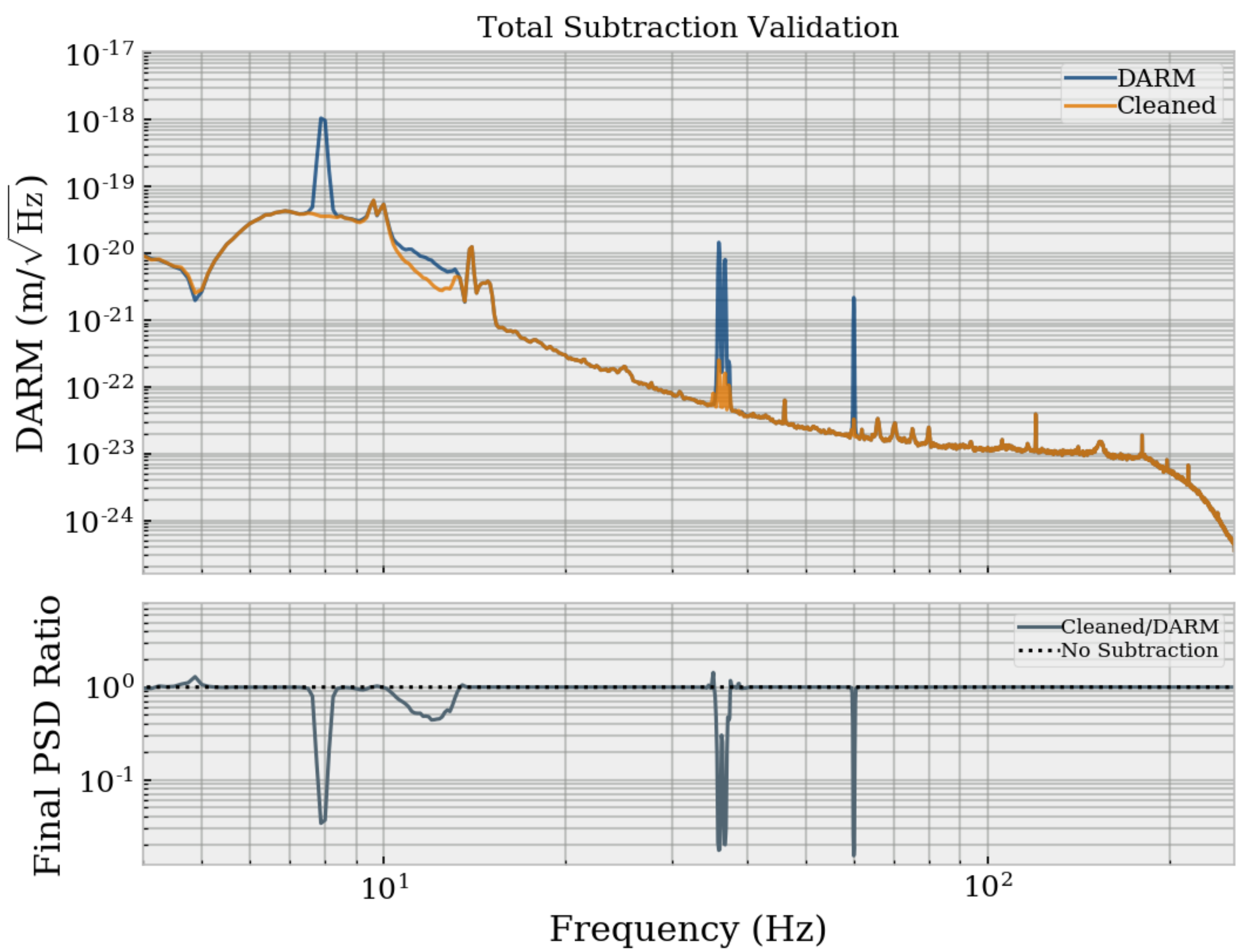


Removing low frequency bilinear jitter added to **mock data** (generated using the MockData repo on GitLab)

LHO O2 CALIBRATION LINES

O2 Channel List

GDS-CALIB_STRAIN
PSL-DIAG_BULLSEYE_PIT_OUT_DQ
PSL-DIAG_BULLSEYE_YAW_OUT_DQ
PSL-DIAG_BULLSEYE_WID_OUT_DQ
MC-WFS_A_DC_PIT_OUT_DQ
IMC-WFS_B_DC_PIT_OUT_DQ
IMC-WFS_A_DC_YAW_OUT_DQ
IMC-WFS_B_DC_YAW_OUT_DQ
ASC-DHARD_P_OUT_DQ
ASC-DHARD_Y_OUT_DQ
ASC-CHARD_P_OUT_DQ
ASC-CHARD_Y_OUT_DQ
LSC-CAL_LINE_SUM_DQ
LSC-SRCL_IN1_DQ
LSC-MICH_IN1_DQ
LSC-PRCL_IN1_DQ
PEM-EY_MAINSMON_EBAY_1_DQ
PEM-EY_MAINSMON_EBAY_2_DQ
PEM-EY_MAINSMON_EBAY_3_DQ
CAL-CS_LINE_SUM_DQ
CAL-PCALY_TX_PD_OUT_DQ
CAL-PCALY_EXC_SUM_DQ
SUS-ETMY_L3_CAL_LINE_OUT_DQ

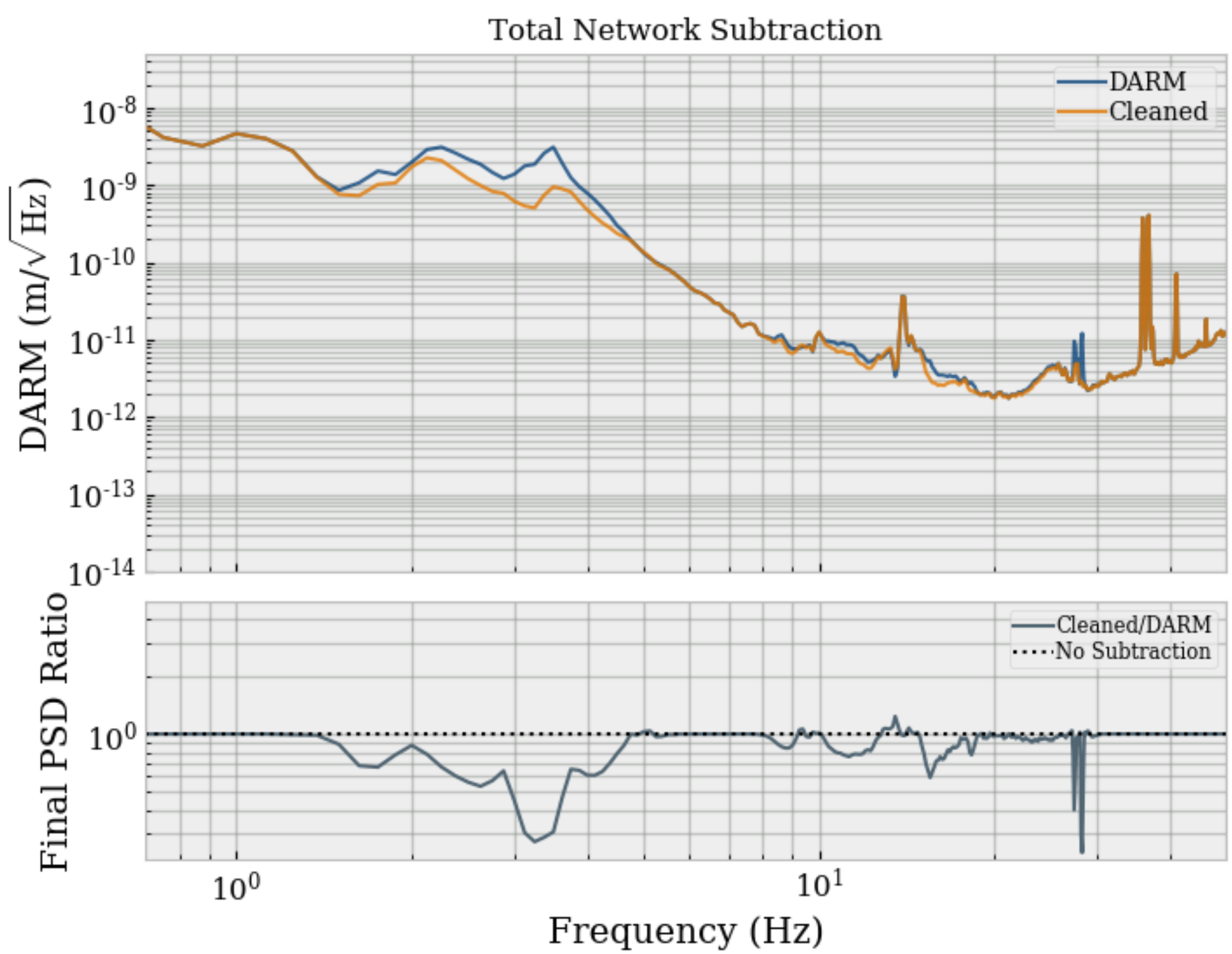


Removing calibration lines and 60 Hz mains from LHO during O2 (+ more?)

LHO O1 - GW150914

O1 Channel List

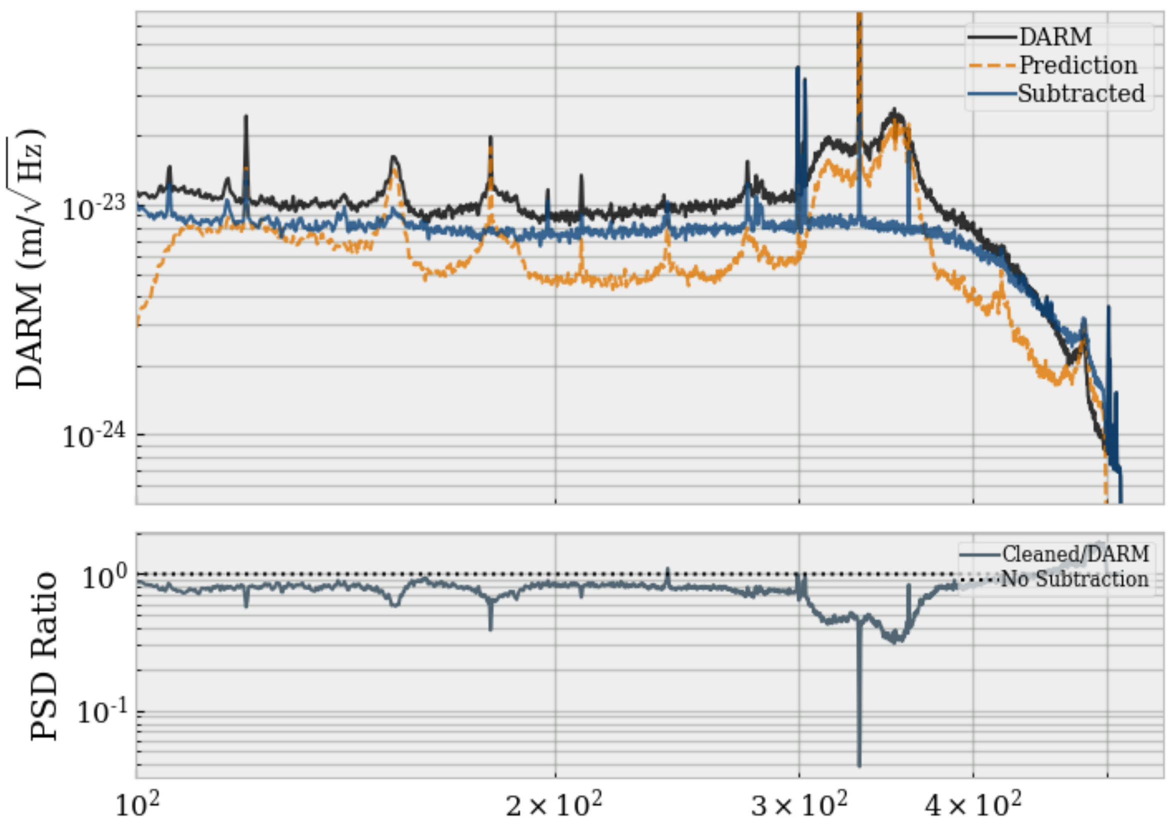
CAL-DELTAL_EXTERNAL_DQ
ASC-CHARD_P_OUT_DQ
ASC-CHARD_Y_OUT_DQ
ASC-DHARD_P_OUT_DQ
ASC-DHARD_Y_OUT_DQ
LSC-MICH_OUT_DQ
LSC-PRCL_OUT_DQ
LSC-SRCL_OUT_DQ
PEM-CS_ACC_HAM4_SR2_X_DQ
PEM-CS_ACC_HAM6_OMC_X_DQ
PEM-CS_ACC_HAM2_PRM_Y_DQ
PEM-CS_MIC_LVEA_INPUTOPTICS_DQ
PEM-CS_MIC_LVEA_OUTPUTOPTICS_DQ
ASC-X_TR_A_PIT_OUT_DQ
ASC-X_TR_A_YAW_OUT_DQ
ASC-X_TR_B_PIT_OUT_DQ
ASC-X_TR_B_YAW_OUT_DQ
ASC-Y_TR_A_PIT_OUT_DQ
ASC-Y_TR_A_YAW_OUT_DQ
ASC-Y_TR_B_PIT_OUT_DQ
ASC-Y_TR_B_YAW_OUT_DQ



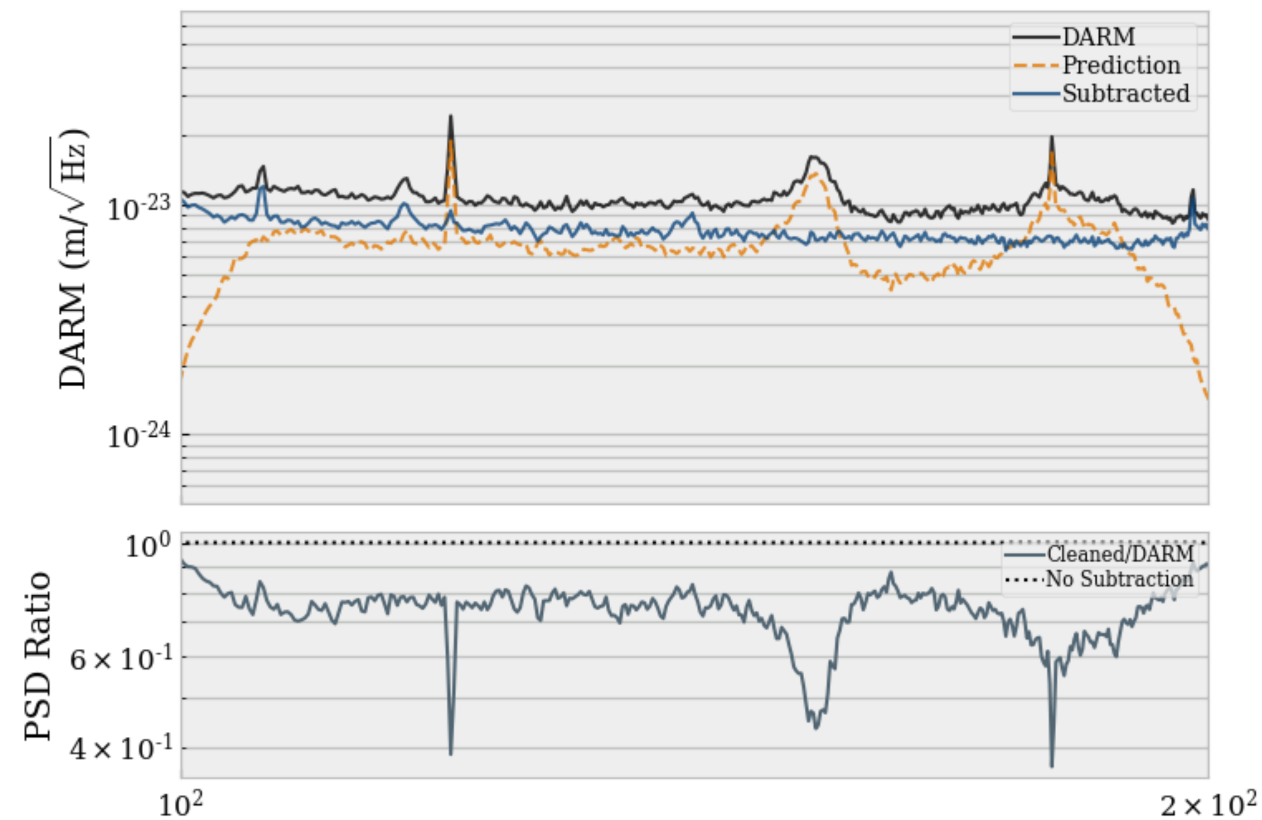
Jitter subtraction at LHO using 1024s of data surrounding GW150914

LHO O2 - BROADBAND JITTER

100-512 Hz Subtraction

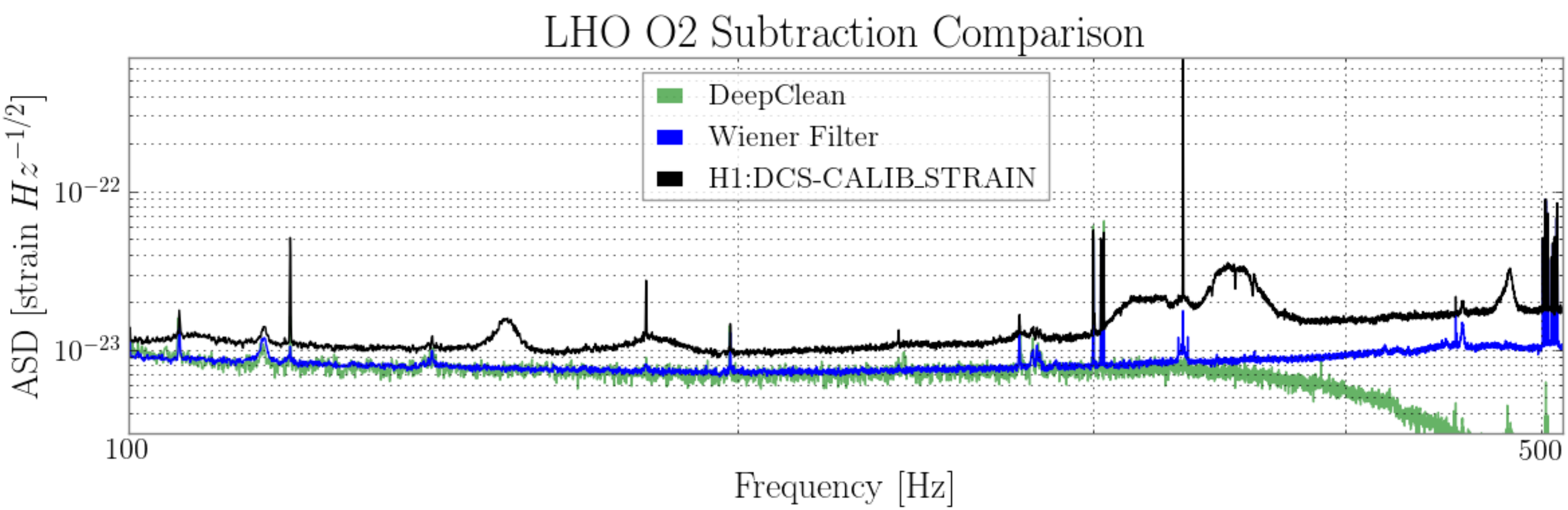


Bandpassed from 100-200 Hz



LHO O2 Linear Noise with "O2 Channel List" (Slide 15).
Bandpassing 100-200 Hz gives better results (less to train)

COMPARISON TO WIENER FILTER

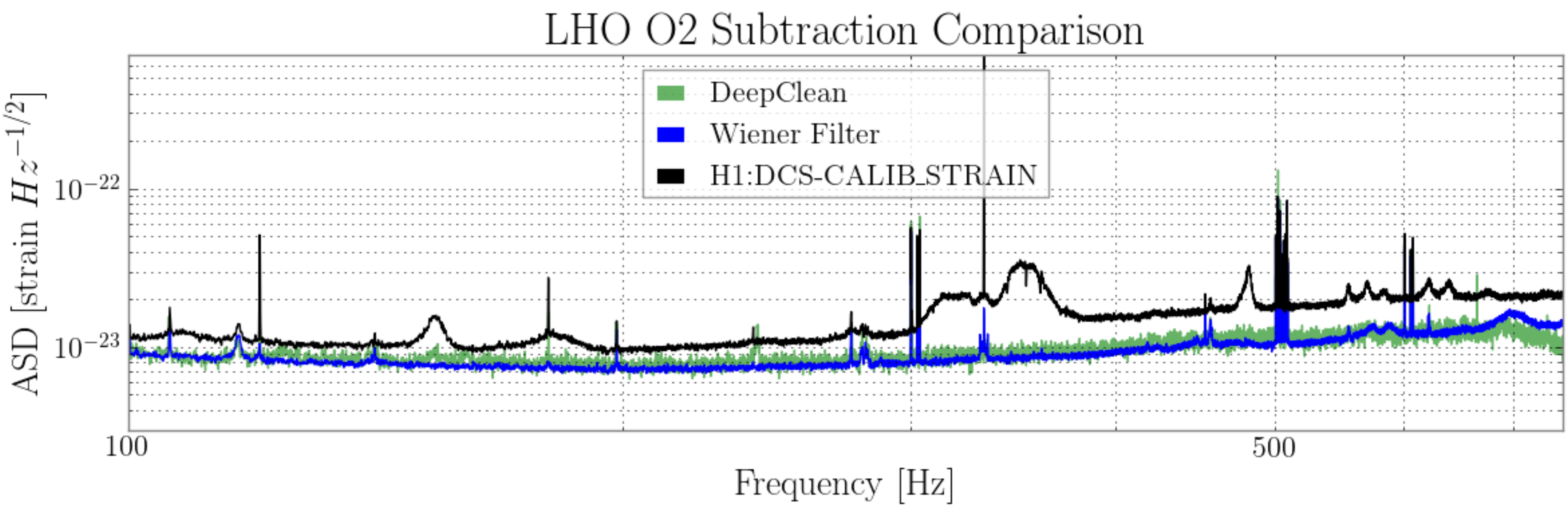


The broadband performance of DeepClean against the O2 linear subtraction is essentially **identical.**

Validates the WF method and DeepClean's network.

Due to **sample_rate** = 1024 Hz, the amplitude falls off as we approach the Nyquist frequency.

COMPARISON TO WIENER FILTER



Same network as previous slide, but with **sample_rate = 2048**.

The network isn't fully converged yet (twice as much data would need to run a little longer) but the performance is still roughly the same.

NEXT STEPS

ATTACK 10-80 HZ BAND

- With the right channels, we will get the expected subtraction (probably after a little tuning), but **what are the *right* channels**? Not obvious!

- If anyone has thoughts about bilinear (or higher order) channel couplings to investigate, please get in touch!

PARALLEL THE WF ANALYSIS

Demonstrate validity though:

- Reproducing the Wiener filter analysis plot by plot with DeepClean

- Repeatability / robustness with varied networks, gps times, IFOs etc

- Determining where and why we beat the linear cleaning (if we do)

TEST IMPROVEMENTS TO PE

- Inject a signal into unclean data. Clean it with DeepClean and perform parameter estimation/ waveform recovery

- Compare SNR before and after cleaning

- Calculate change in detectable volume, $\langle VT \rangle$ in given frequency band

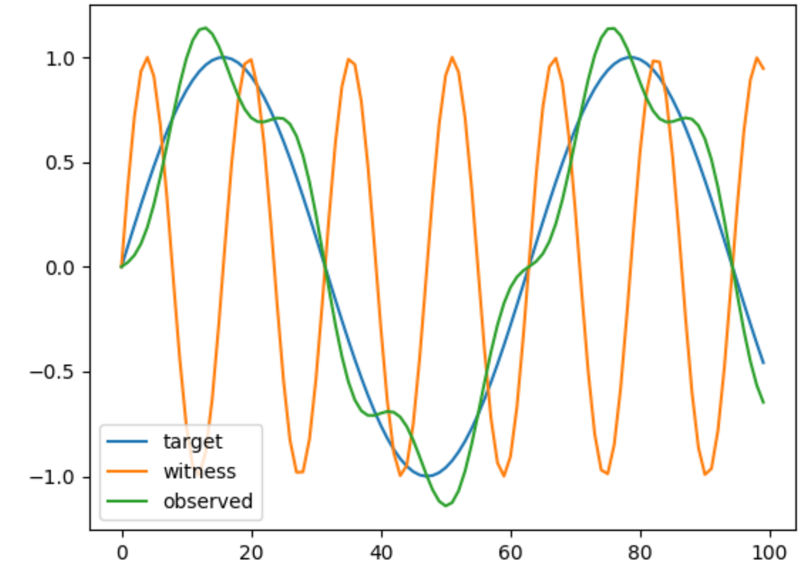
THANK YOU

Contact

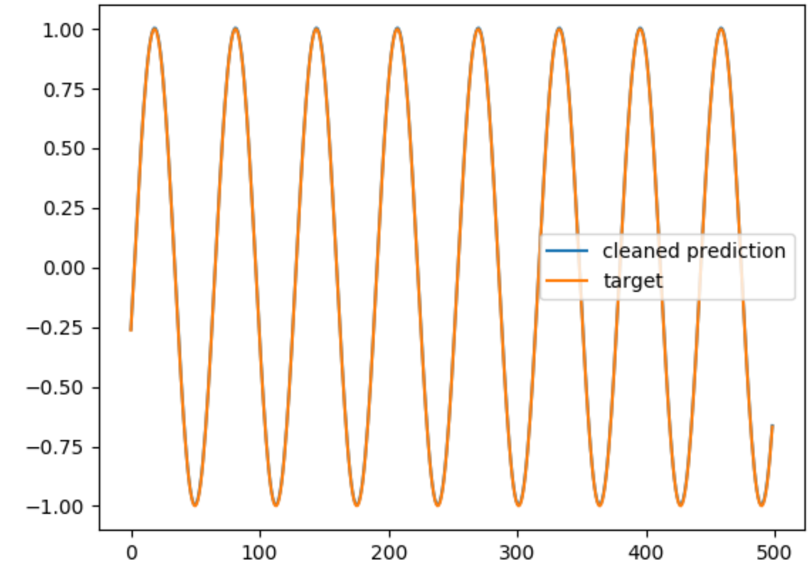
Rich Ormiston: ormiston@umn.edu
Michael Coughlin: michael.w.coughlin@gmail.com

EXTRA SLIDES

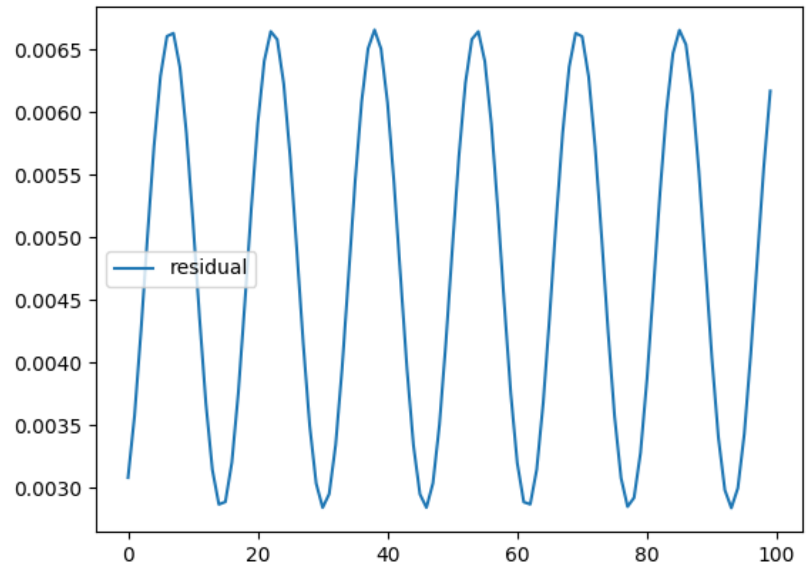
EX: INFORMAL TARGET - I



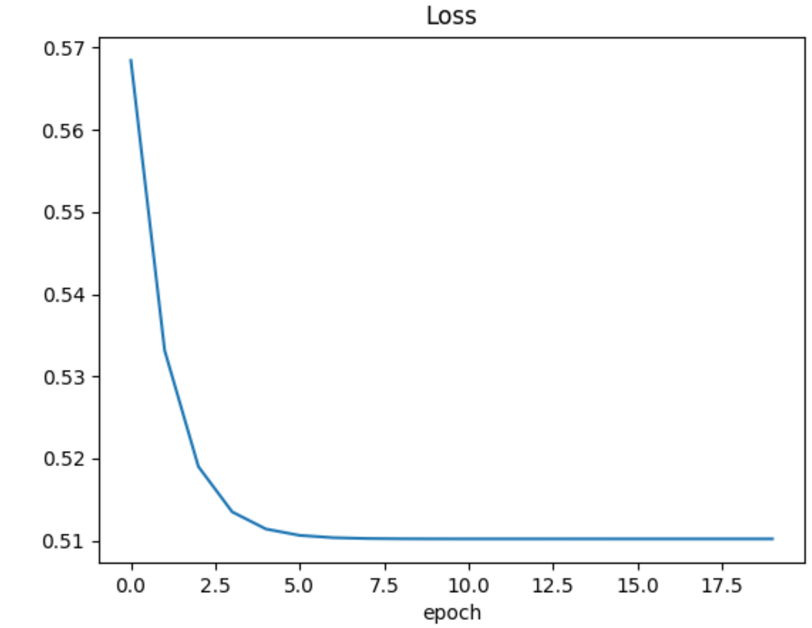
Input data. We never see the real target, just what is observed



Comparing cleaned output versus the actual target



Residual error between cleaned prediction and target



Decreasing and leveling loss demonstrates convergence

EX: INFORMAL TARGET - II

Question:
How does the witness channel "know" to only subtract itself and not to remove any of the true signal?

Answer:
The frequency of the witness is fixed, so the only way to minimize the loss (will never be zero!) is to subtract the contribution of the witness. That is, the network sees

$$\left\{A\sin(\omega_A t + \phi_A) + B\sin(\omega_B t + \phi_B)\right\} - B'\sin(\omega_B t + \phi_{B'})$$

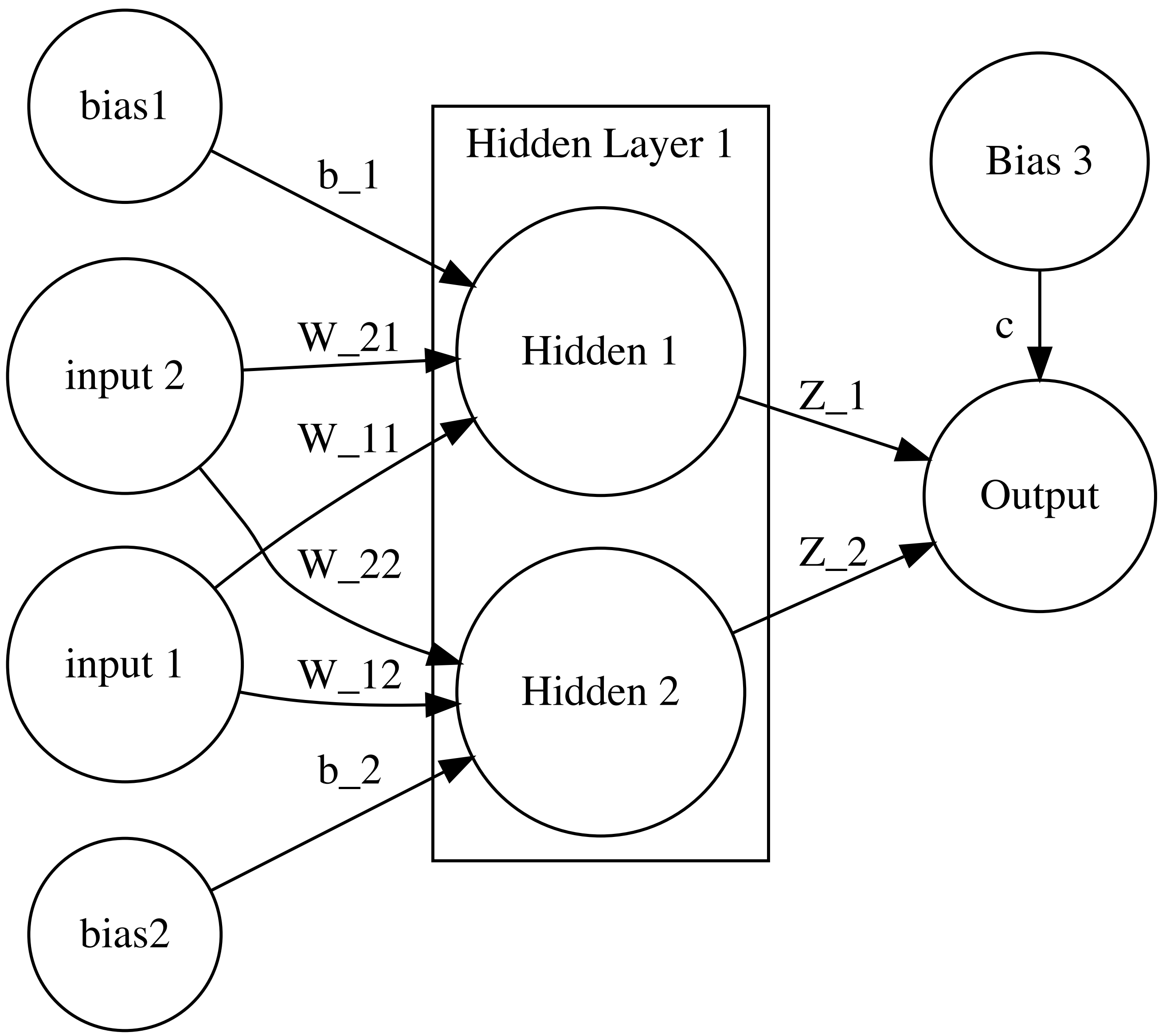
Where {...} is the observed signal and the last term is the witness. Only B' and $\phi_{B'}$ are tunable. The minimal loss is obtained when the network learns to set B'=B and $\phi_B = \phi_{B'}$.

EX: INFORMAL TARGET - III

Conclusion - the network will only learn how to subtract witness channels (or nonlinear combinations thereof) and will leave behind the underlying signal(s).

For more examples of this, look here

SAMPLE NETWORK



$$= Z^T(W^T\mathbf{i} + \mathbf{b}) + c$$

$$\text{More generally, Output} = g\Big(Z^T f(W^T\mathbf{i} + \mathbf{b}) + c\Big)$$