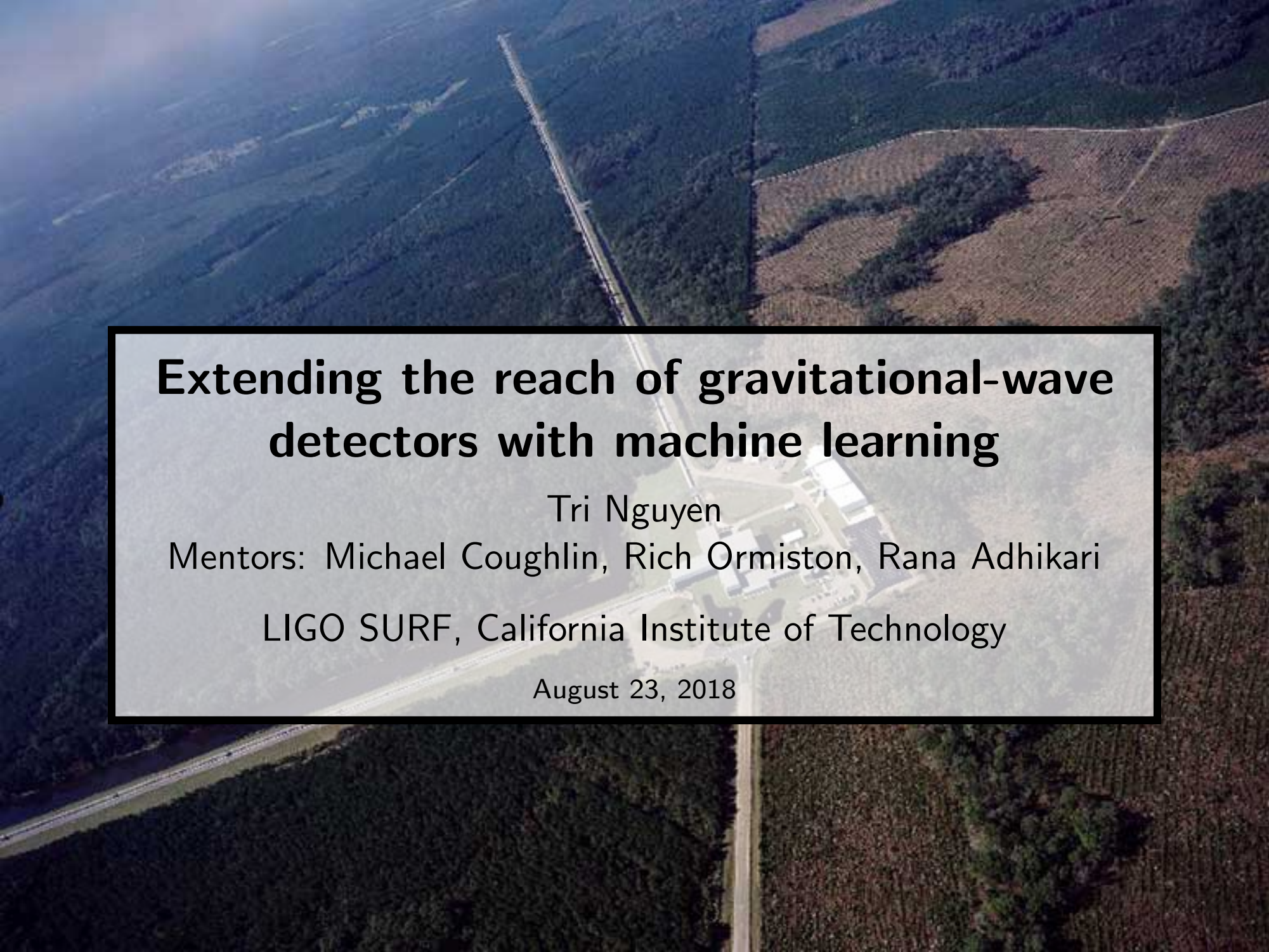


Extending the reach of gravitational-wave detectors with machine learning

Tri Nguyen
Mentors: Michael Coughlin, Rich Ormiston, Rana Adhikari

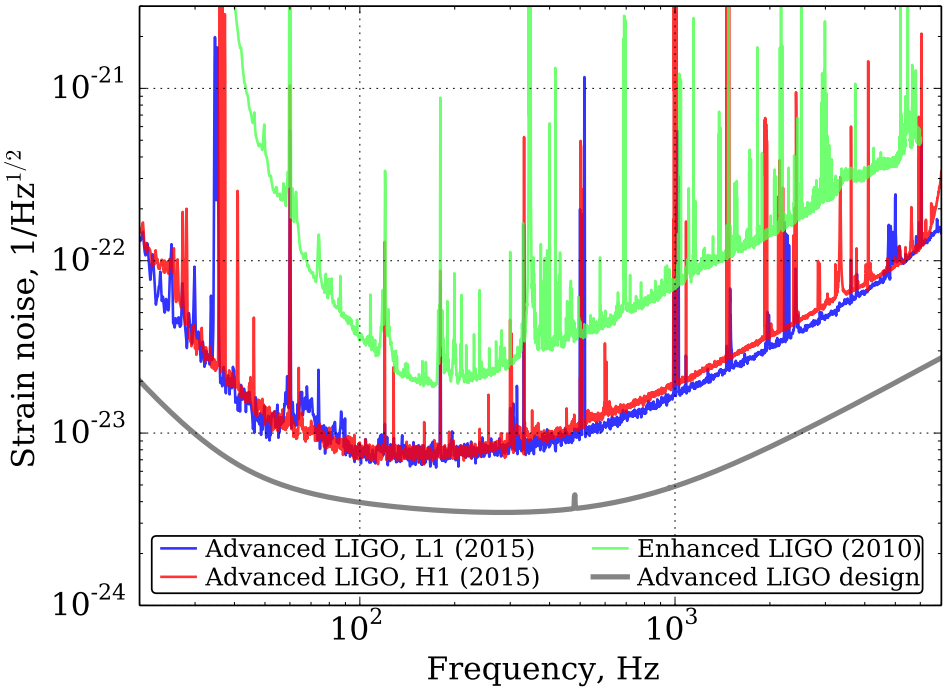
LIGO SURF, California Institute of Technology

August 23, 2018

Table of Contents

The sensitivity of Advanced LIGO

- Advanced LIGO, or aLIGO, has detected gravitational waves from many black-hole and neutron-star mergers.
- However, signals from many of these objects still lie below the detector sensitivity limit.
- An improvement in LIGO's sensitivity would allow us to uncover these objects and make gravitational-wave detections a more routine occurrence.



The sensitivity of LIGO Livingston (L1) and LIGO Hanford (H1) during the first observation run O1 [1].

Noise regression analysis at LIGO

- LIGO keeps track of its noise sources via the physical environmental monitor (PEM) channels, or **witness channels**.
- A noise source can couple **linearly** or **non-linearly** into the detector **differential arm length** (DARM).
- The current regression method, the Wiener-Kolmogorov filter, fails to remove the non-linear contributions.
- Characterizing and filtering out these non-linear noises is challenging because the coupling mechanisms are sophisticated.
- Despite these complexity, a **neural network** may be able to learn them.

Table of Contents

Fundamental v. Non-fundamental noises

We can categorize the noise sources into:

- **Fundamental noises**:
  - Define LIGO's baseline sensitivity limit.
  - Can only be reduced by improving the detector design.
  - Example: quantum noise, thermal noise, etc.
- **Non-fundamental noises**:
  - Instrumental and environmental effects.
  - Can be subtracted given there are witness channels monitoring them.
  - Example: seismic noise, magnetic noise, calibration lines, beam jitter, suspension noise, etc.

Goal: predict and subtract the non-fundamental noises while keeping the signals and the fundamental noises intact.

Mathematical formulation

Given a witness channel w_i at time t, the DARM output is:

$$h(t) = h_s(t) + \mathcal{T}[w_i(t)]$$

where:

h_s is the gravitational-wave signal.

$\mathcal{T}$ is the **transfer function**, linear or non-linear, coupling the witness channel and the DARM.

The network will predict a transfer function $\tilde{\mathcal{T}}$ and the DARM $\tilde{h}(t')$ at time $t' > t$:

$$\tilde{h}(t') = \tilde{\mathcal{T}}[w_i(t')]$$

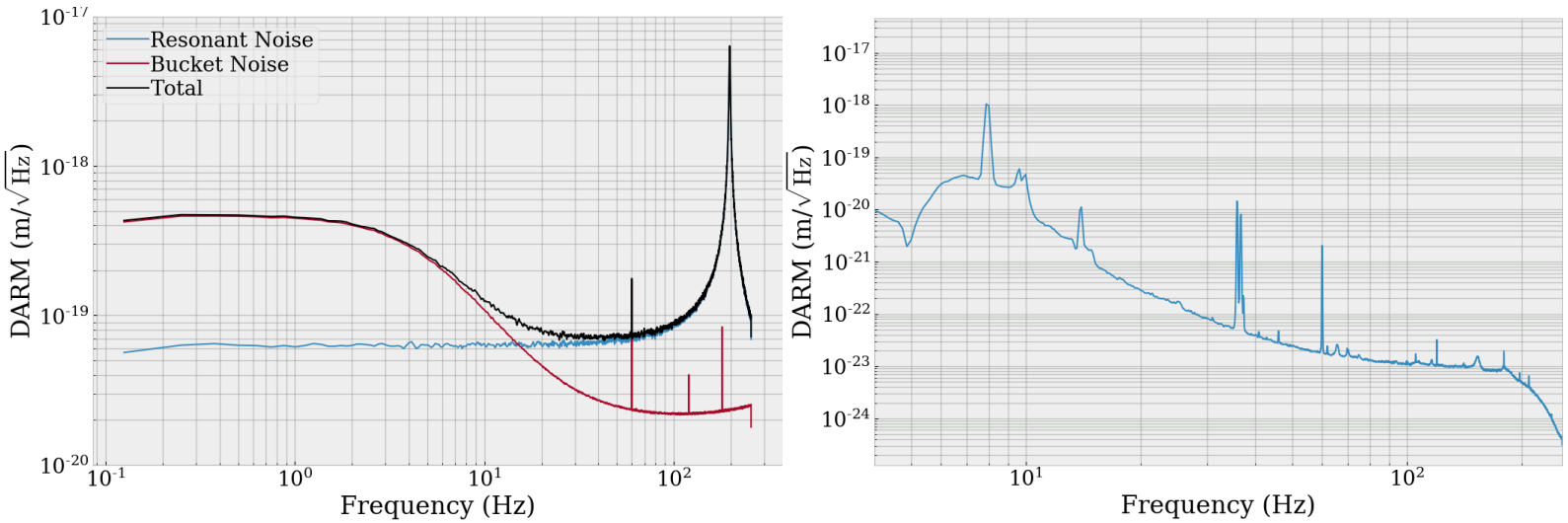
Goal: minimize the **mean square error** (MSE) or the **mean absolute error** (MAE) between h and $\tilde{h}$ at $t' > t$.

Table of Contents

Mock v. Real data

- We perform our analysis on both mock and real data.
- While mock data provide a reliable gauge to measure the network capacity, ultimately we want to test on real data.
- To generate mock data, we couple white noises into the DARM via the **resonance function**:

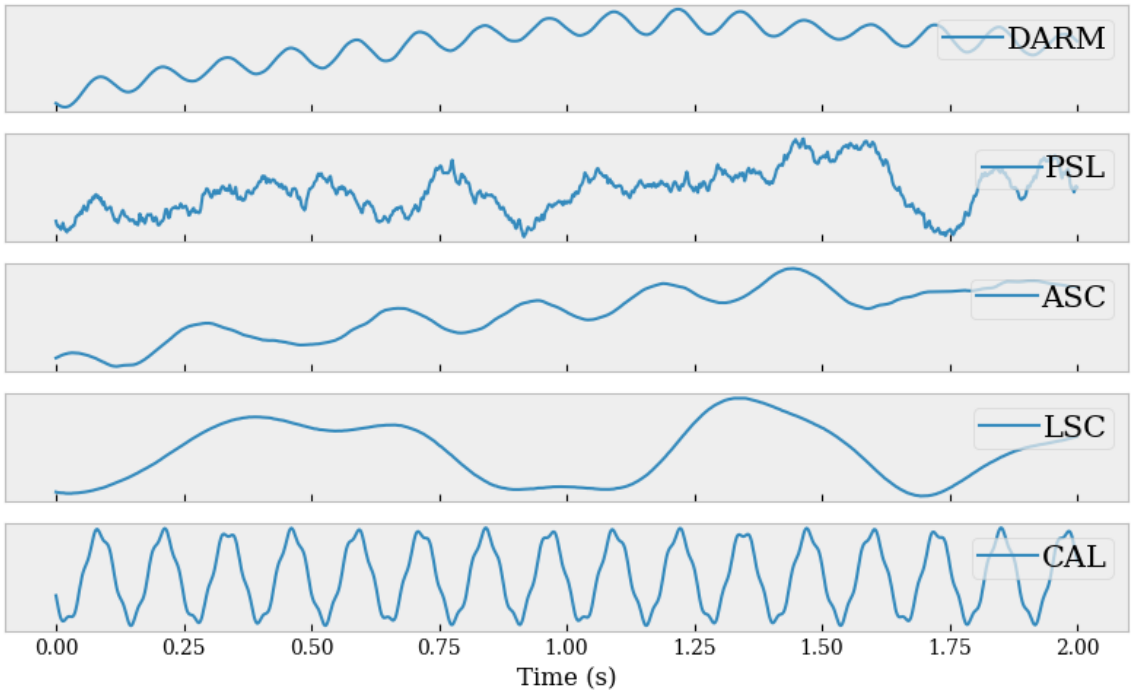
$$y(w) = \frac{x(w)}{w_0^2 - w^2 + i\frac{w_0 w}{Q}}$$



Left: Resonance spectra $w_0 = 5$ rad/s and $Q = 100$. Right: DARM spectra from LIGO Hanford on August 14, 2017.

Visualizing the data

- The input data consists of multiple witness channels.
- Each witness channel has a duration of 2048 seconds and a sample rate of 512 Hz. The number of samples is 1,048,576.



Sample witness channels by subsystems from LIGO Hanford on August 14, 2017.

Data preprocessing

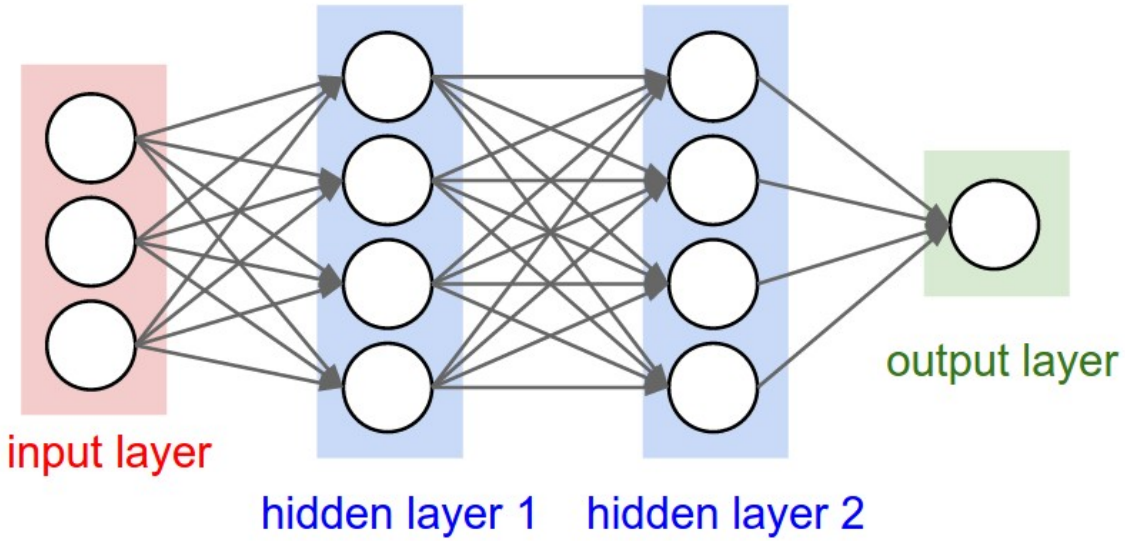
- **Sample-rate conversion**: Up-sample or down-sample so all channels have the same sampling rate of 512 Hz.
- **Standard scaling**: Normalize the mean and standard deviation of each channel to 0 and 1:

$$x_{ij}^{norm} = \frac{x_{ij} - \frac{1}{N}\sum_k x_{kj}}{\sqrt{\frac{1}{N}\sum_k (x_{kj} - \frac{1}{N}\sum_k x_{kj})^2}} \text{ for each } j$$

  where x_{ij} is the ith data point of the witness channel j, and N is the number of samples.
- **Bandpass filter**: Attenuate frequencies outside a certain range.
- **Lookback window**: Divide each channel into short and overlapping series, each consisting of 16 samples, or 0.03125 seconds. After lookback, the dataset has 1,048,546 data examples, each of length 16.

Neural networks: Basics

- Neural networks are a set of non-parametric, supervised machine learning algorithms.
- A neural network may consist of up to few thousands of nodes. Each carries parameters (**weights** and **biases**) to characterize data features.
- Often used in classification, it is capable of recognizing highly complex patterns (e.g. language models, stock market, etc.)



Neural networks block diagram. Refer to [2].

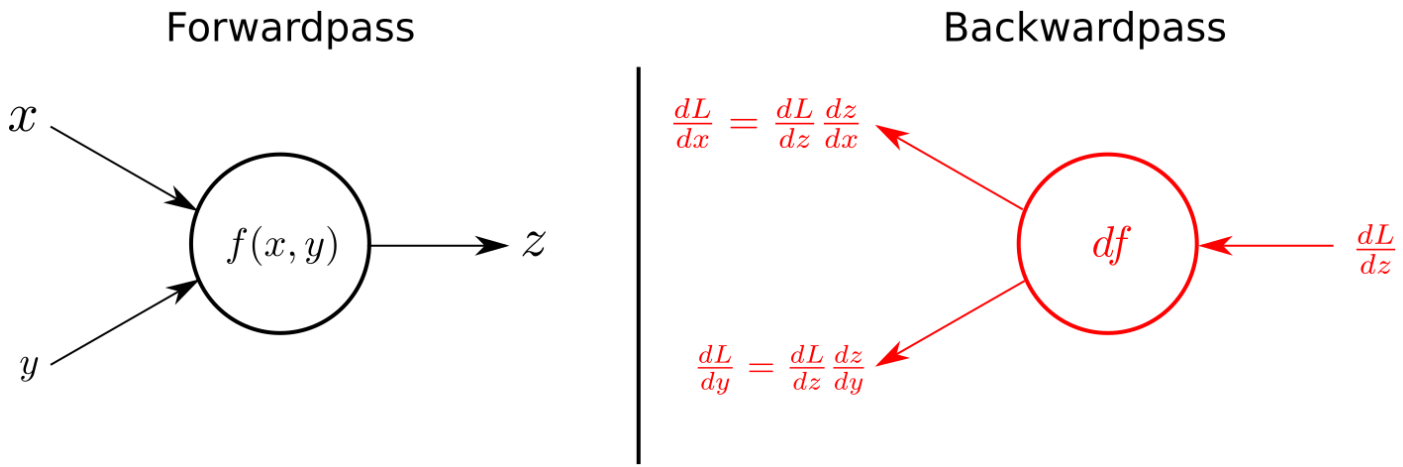
Neural networks: Learning

Neural networks learn by looping over a **training set** and minimizing a **loss function**:

1. **Forward propagation**: Compute the output.
2. **Backward propagation**: Compute the gradients of weights and biases by applying the chain rule recursively.
3. **Gradient descent**: Update the weights and biases:

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \alpha \nabla J(\mathbf{X}_i)$$

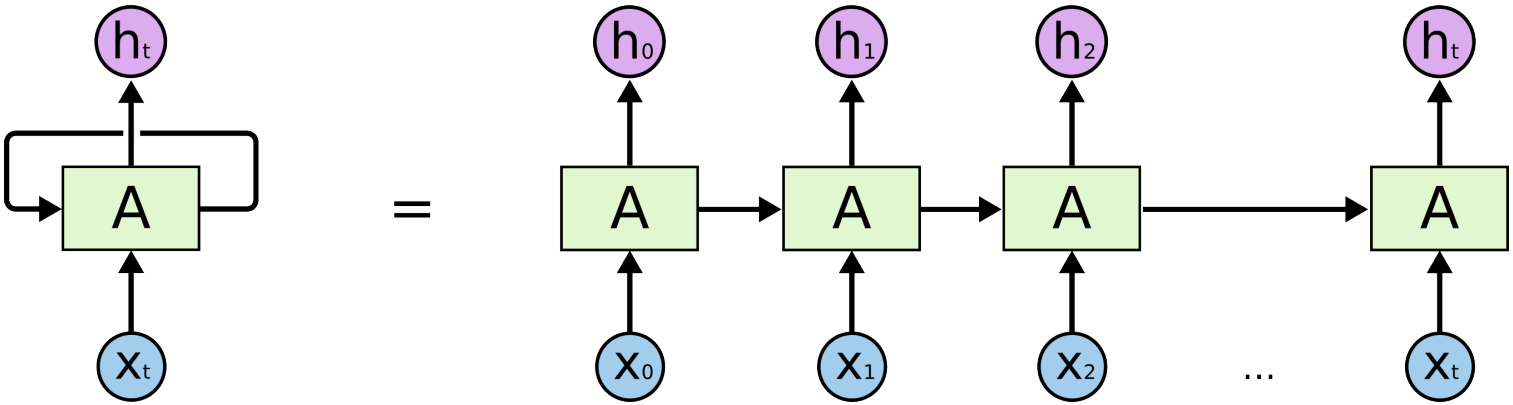
where α is the learning rate and J is the loss function.



Foward and backward pass block diagram [3].

Recurrent neural networks

- A **recurrent neural network** (RNN) is a subclass of neural networks which specializes in processing sequential inputs, like gravitational-wave data.
- A major strength of RNNs over traditional networks is the ability to store and use information from past inputs.
- For our analysis, we use a special type of RNNs called a **Long Short-Term Memory** (LSTM) network.

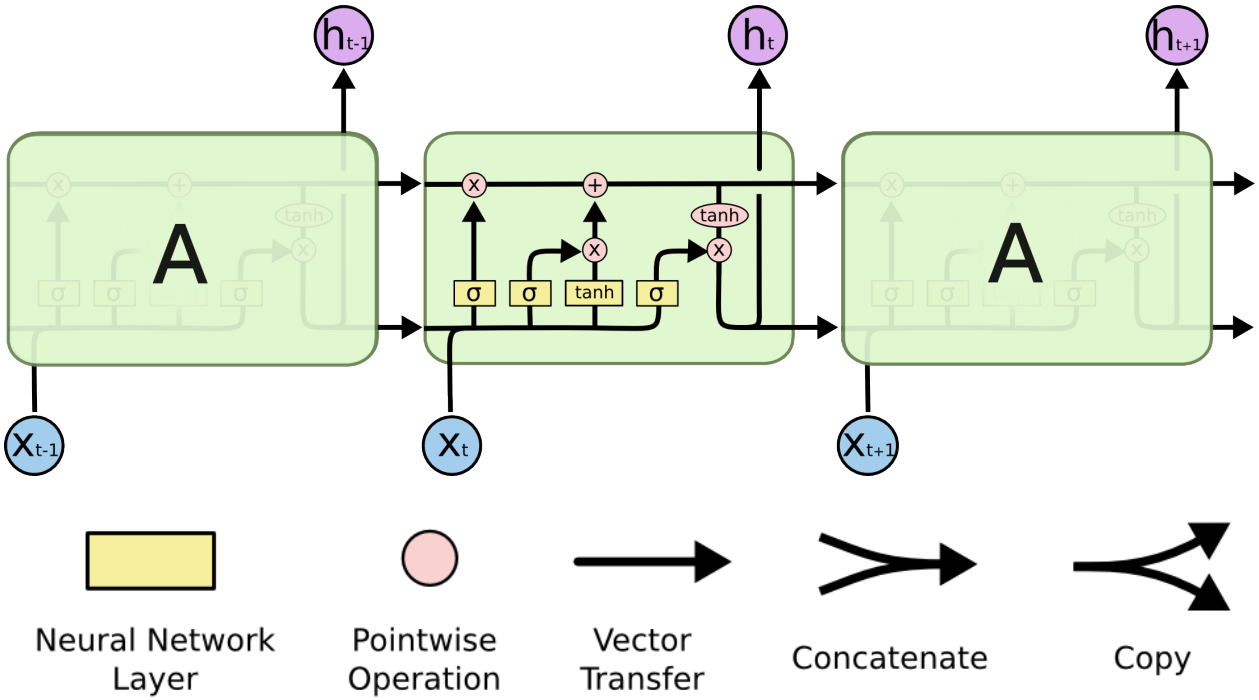


Unrolled RNN structure. Refer to [4].

Long Short-Term Memory networks

The advantages of LSTM networks over RNNs:

- The **cell state** carry information from past inputs ⇒ The network can learn long-term dependencies and eliminate the vanishing (exploding) gradient problem.
- **Gates** let information to be added to (input gates) and removed from (forget gates) the cell state.



A typical LSTM layer contains four interacting sub-layers. Refer to [4].

Current framework: DeepClean

- The current analysis framework, DeepClean[1], consists of five separate LSTM networks.
- Each network has a different set of parameters and runs on a different frequency band. The frequency bands are 3-9 Hz, 10-13 Hz, 20-30 Hz, 30-41 Hz, and 57-63 Hz.
- Each network consists of layers of different type:
  - LSTM: Return a sequence which get fed into the next LSTM layer.
  - Dense: Followed the LSTM layers. Predict the DARM.
  - Batch normalization: Followed each LSTM and Dense layer (except for the output layer). Normalize the mean and standard deviation across each batch to 0 and 1.
- Each network has in total 5869 trainable parameters (weights and biases).

[1] Available at https://git.ligo.org/rich.ormiston/DeepClean

Hyperparameter tuning

- **Hyperparameters** determine the network structure (e.g. number of LSTM layers) and govern the learning process (e.g. learning rate).
- They cannot be learned from data. Different problems require different sets of hyperparameters.
- Finding the optimal hyperparameters requires experimenting with different networks and evaluate their relative performance $\Rightarrow$ computationally expensive.
- General procedure:
  1. Pick a simple network architecture and increase complexity as needed.
  2. Overfit a small training dataset $\Rightarrow$ the network is capable of learning the transfer function.
  3. Reduce the overfitting by choosing the optimal learning parameters.

Hyperparameter tuning

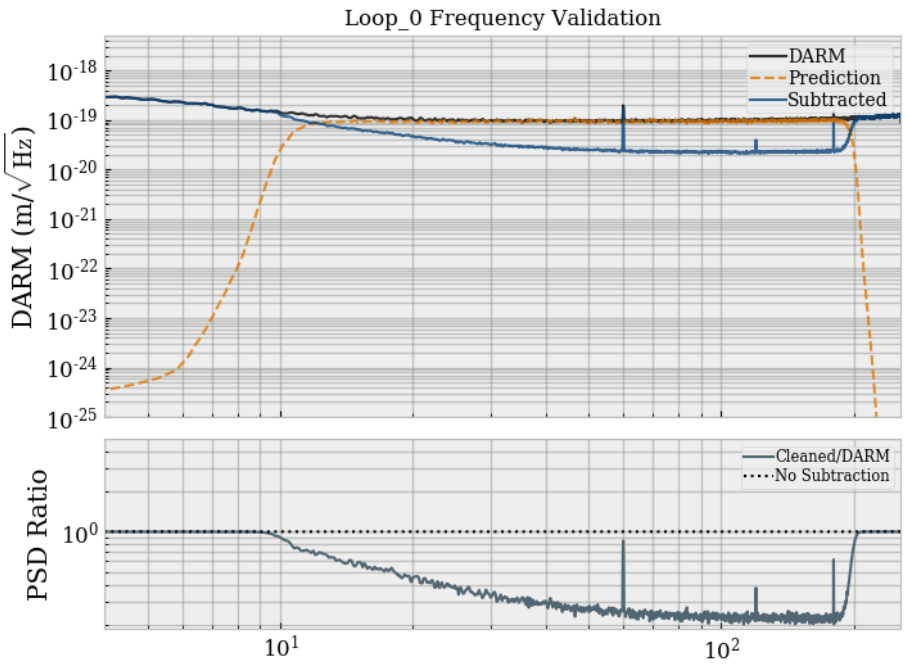
Picking the best learning parameters

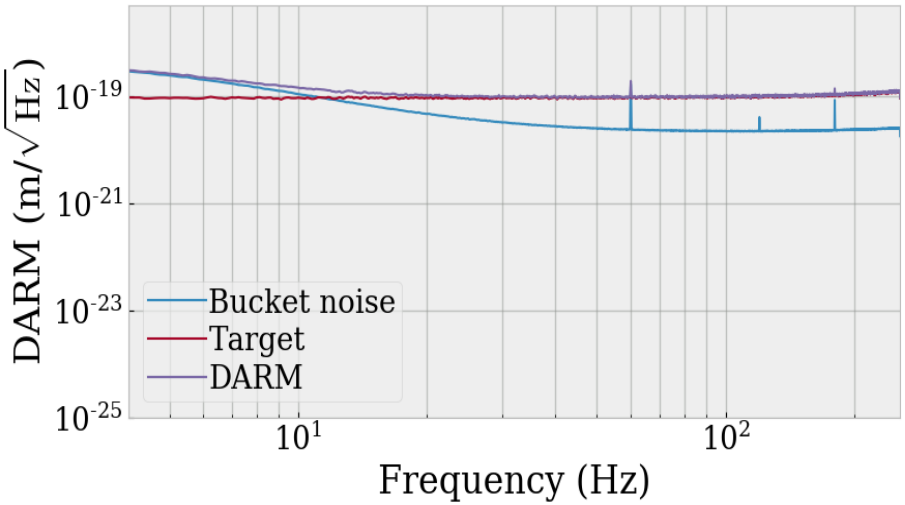
- We apply **holdout cross-validation**: partition data into a training and a validation set. After each iteration through the training set, the network is tested on the validation set.
- To search for the optimal parameters, we apply these algorithms:
  - **Random search**: Randomly sample the hyperparameters given some prior guesses.
  - **Tree-structured Parzen estimators**: Divide past evaluations into two groups: the best parameters and the rest. Pick the parameter belong most likely to the first group and less likely to the second group by constructing a likelihood ratio.
  - **Gaussian process**: Assume the loss function is a Gaussian stochastic function. Predict the loss by constructing posterior distribution given a prior distribution and past evaluations.

Table of Contents

Results: Mock data

When tested on the resonant noise with $w_0 = 13.15$ rad/s and $Q = 1000$, the network successfully subtracted the noise and left out the 60-Hz AC power line and the rest of the bucket noise.

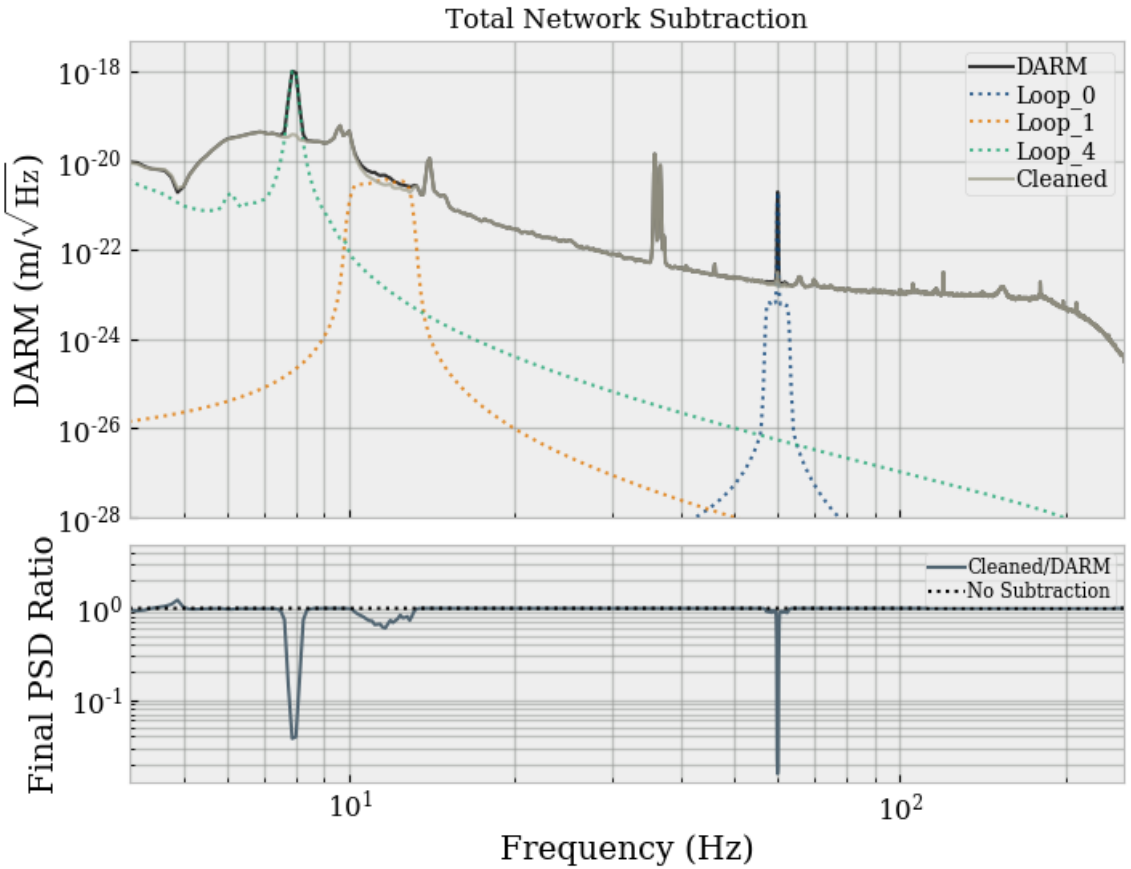




Left: The subtraction in the 10-250 Hz band. Right: The expected outcome. The spectra is dominated by the resonant noise.

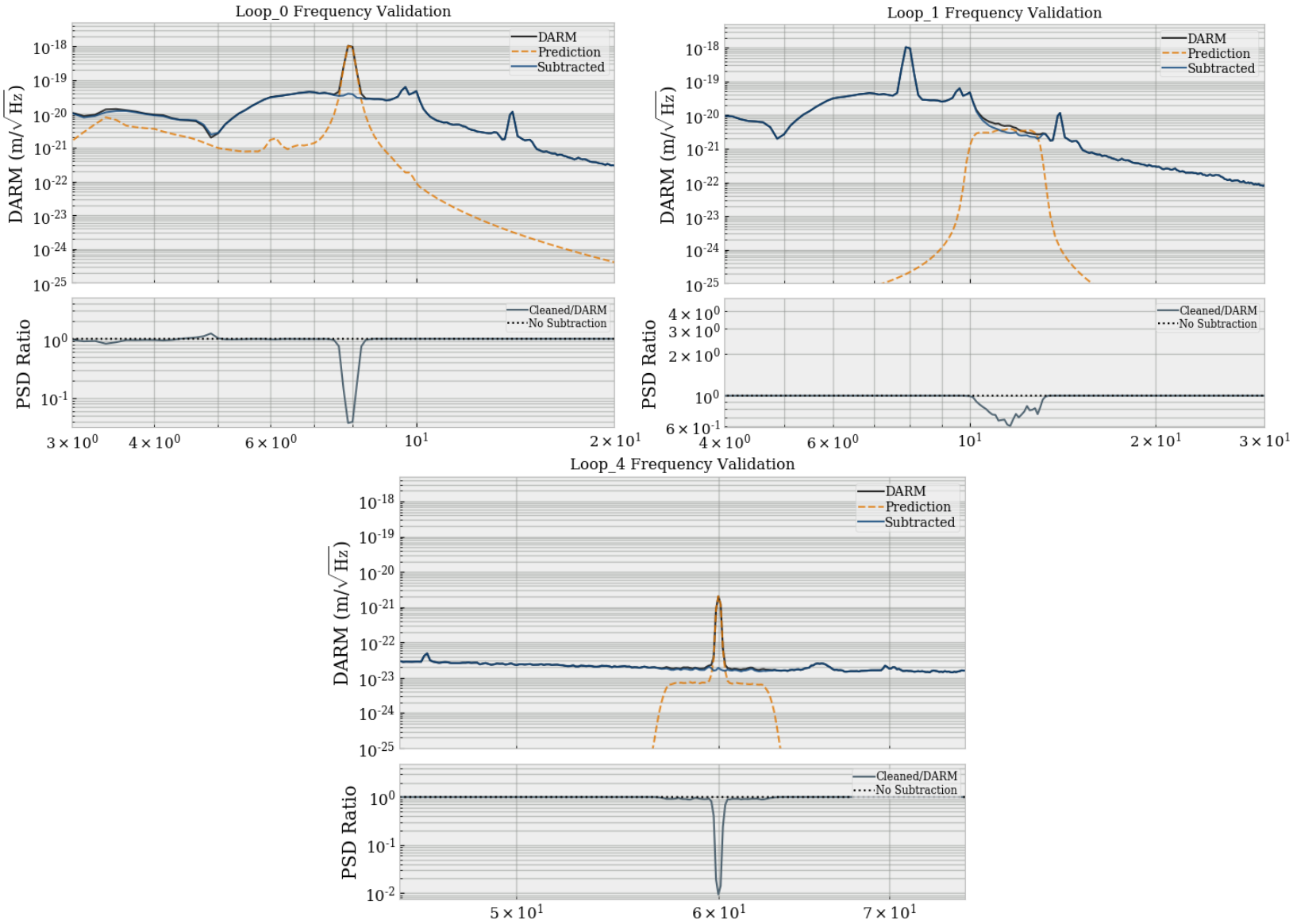
Results: Real data

When tested on LIGO Hanford data, the network successfully subtracted the calibration lines at 7 Hz and 60 Hz.



Subtraction results on LIGO Hanford data on August 14, 2017.

Results: Real data

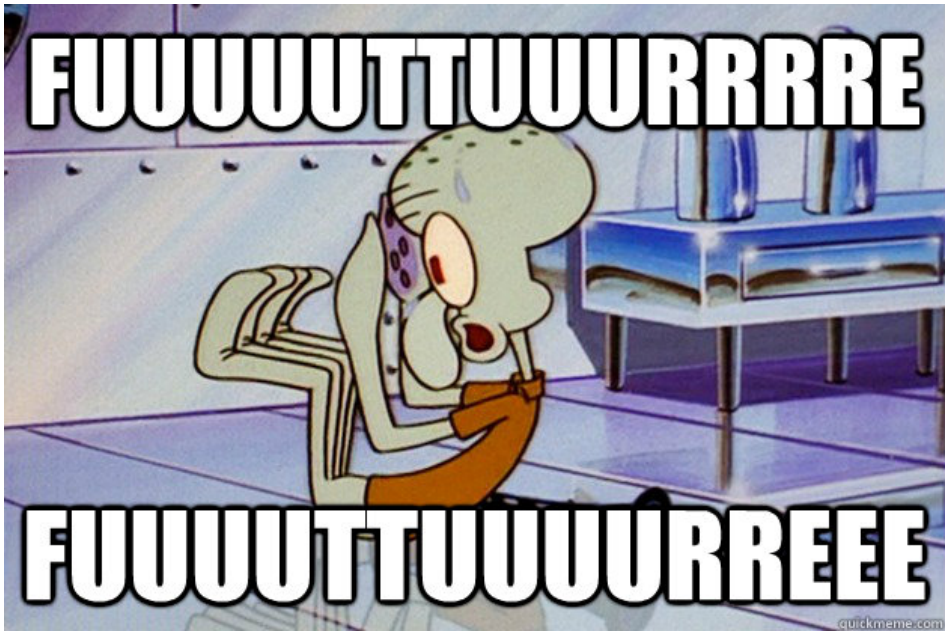


Subtraction results on the band 3-9 Hz (top left), 10-13 Hz (top right), and 57-63 Hz (bottom).

Table of Contents

Future works

- Further study and tuning is needed before DeepClean can be applied to real-time noise filtering.
- Not all witness channels carry the same weight. Ultimately, we would like to understand which and why a channel is important.



Special Thanks

Michael Coughlin, Rich Ormiston, Rana Adhikari

Alvin, Vinny, Sky, and the LIGO SURF group

The LIGO SURF program

References I

[1] Benjamin P. Abbott et al.
Sensitivity of the Advanced LIGO detectors at the beginning of gravitational wave astronomy.
Phys. Rev., D93(11):112004, 2016.
[Addendum: Phys. Rev.D97,no.5,059901(2018)].

[2] Andrej Karpathy.
Neural networks part 1: Setting up the architecture, November 2017.

[3] Frederik Kratzert.
Understanding the backward pass through Batch Normalization Layer, 2016.

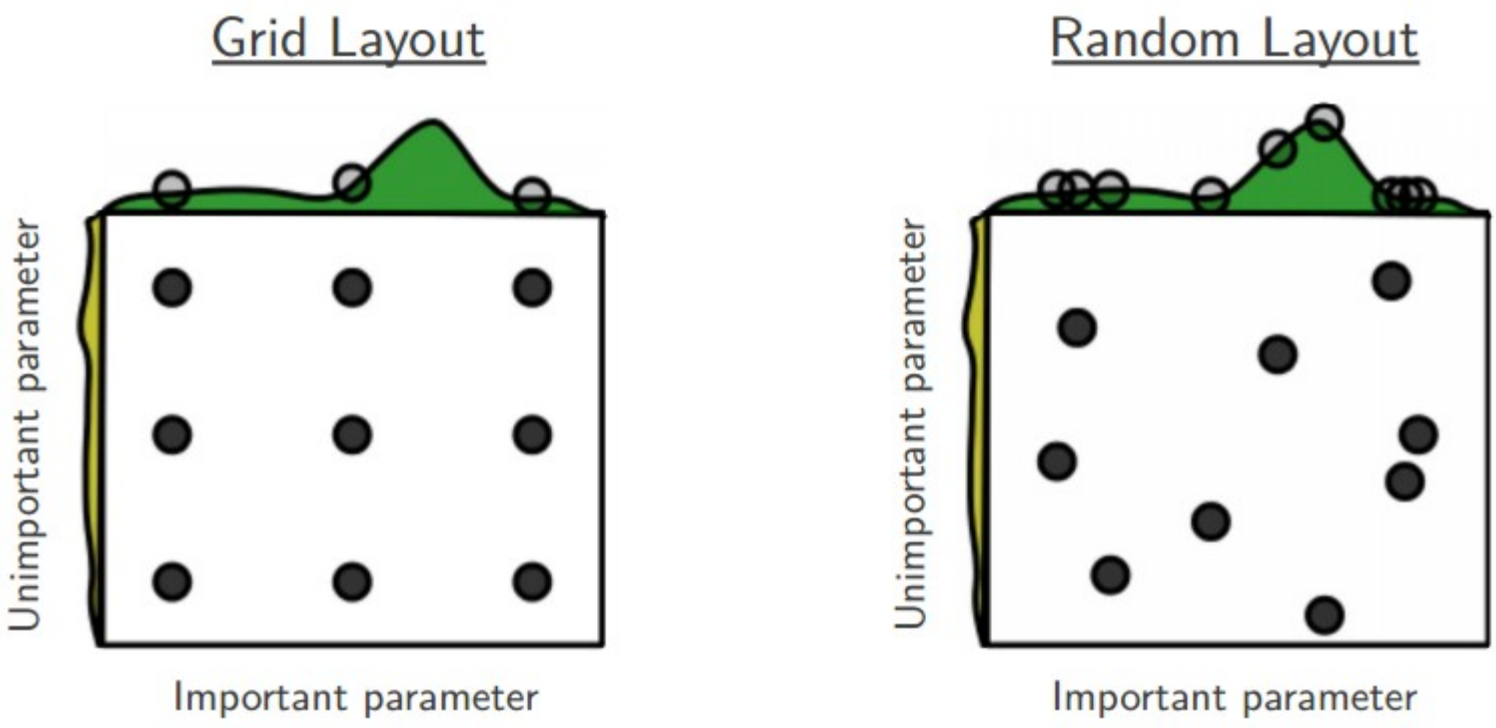
[4] Christopher Olah.
Understanding LSTM Networks, 2015.

References II

[5] James Bergstra and Yoshua Bengio.
Random search for hyper-parameter optimization.
JMLR, 13, 2012.

[6] Yurii Shevchuk.
Hyperparameter optimization for Neural Networks, 2016.

Extra Slide: Grid v. Random search



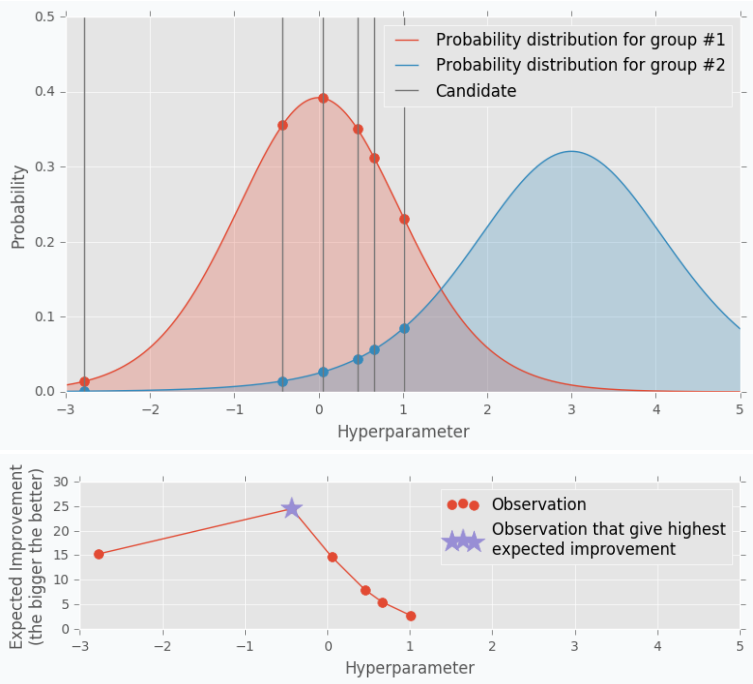
Random search explores the parameter space more efficient than grid search. Refer to [5].

Extra Slide: TPE

TPE picks the parameter x with the highest expected improvement:

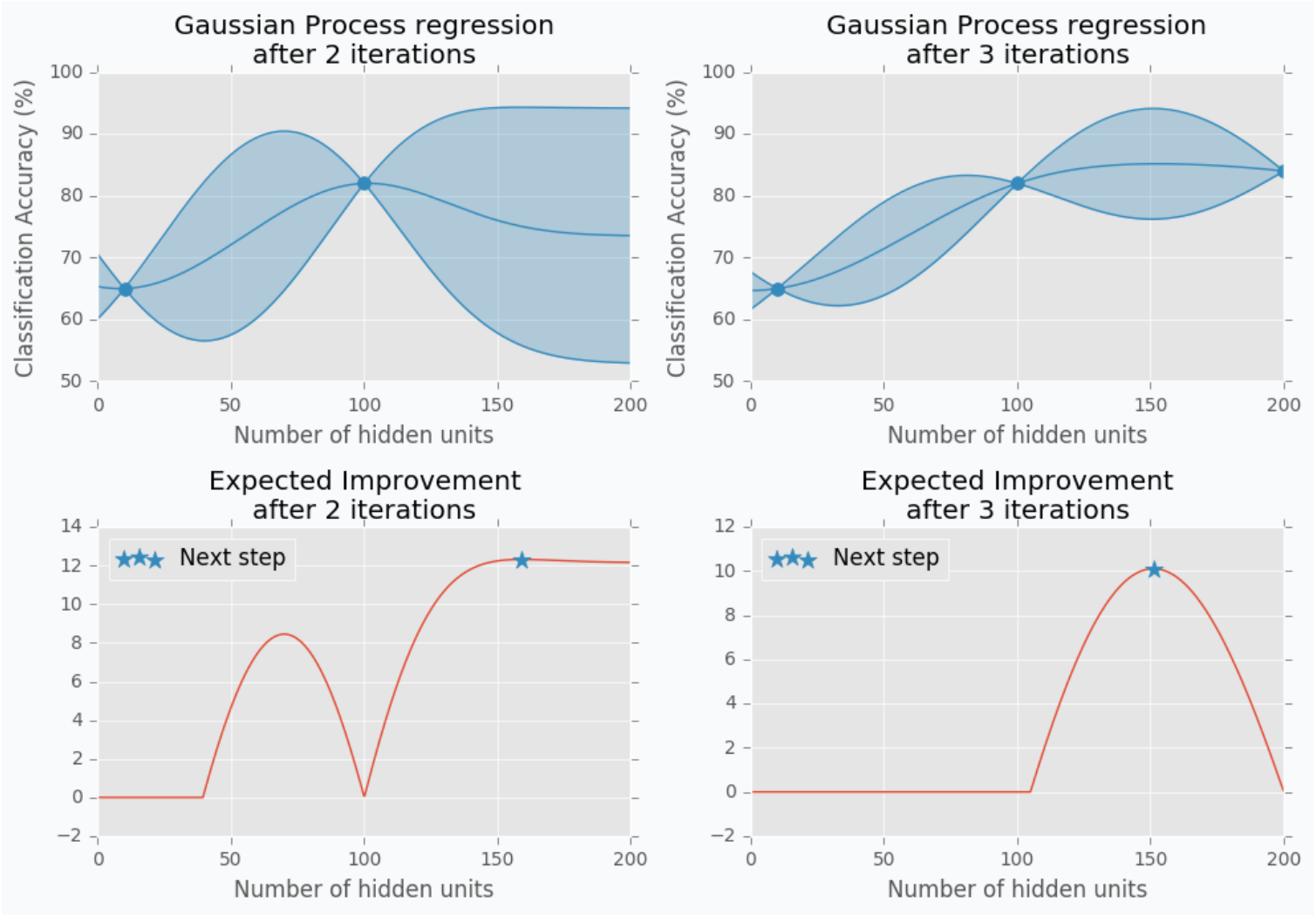
$$r(x) = \frac{l(x|D)}{g(x|D)}$$

where $l(x|D)$ and $g(x|D)$ are the likelihoods of x belong to first and second groups.



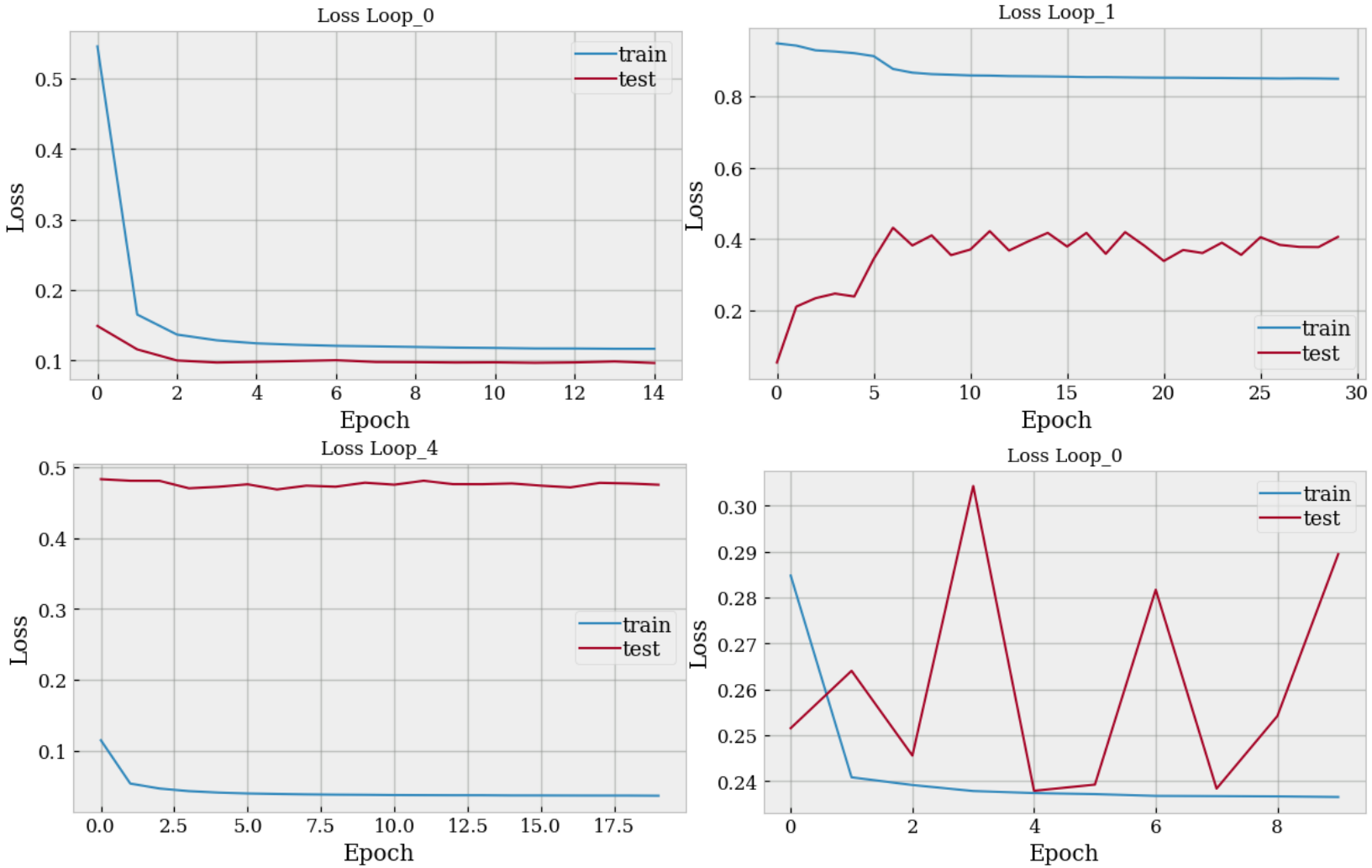
TPE predicts the best parameters by constructing a likelihood ratio of the two groups. Refer to [6].

Extra Slide: Gaussian process



The Gaussian process algorithm predicts the number of hidden units that will maximize the validation accuracy of a Dense network. Refer to [6].

Extra Slide: Loss Curves



Loss curves of Loop 0 (top left), Loop 1 (top right), Loop 4 (bottom left), and resonance Loop 0 (bottom right).