A. Iess

# Machine Learning for noise cancellation in systems with non-linear couplings

# ML for noise cancellation in systems with non-linear coupling

## KEYWORDS

- Gravitational wave interferometer
- Non-linear couplings in gravitational wave detectors
- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory (LSTM)

# ML for noise cancellation in systems with non-linear coupling

## Detector Noise

- Large number of noise sources
- Non-stationary spectrum
- Large number of degrees of freedom

↓

AFFECTS LOCK ACQUISITION

Traditional approach involves filtering the raw (*e.g.* Wiener Filter):

$$\vec{y} = \vec{w}^{\mathrm{T}}\vec{x}$$

$$\vec{e} = \vec{d} - \vec{y}$$

$$\xi \equiv E[\vec{e}^{\,2}] = E[\vec{d}^{\,2}] - 2\vec{w}^{\mathrm{T}}\vec{p} + \vec{w}^{\mathrm{T}}R\vec{w}$$

$$\frac{d\xi}{dw_i} = 0 \qquad R\vec{w}_{\mathrm{optimum}} = \vec{p}$$

Minimize error by optimizing TF function of witness sensors

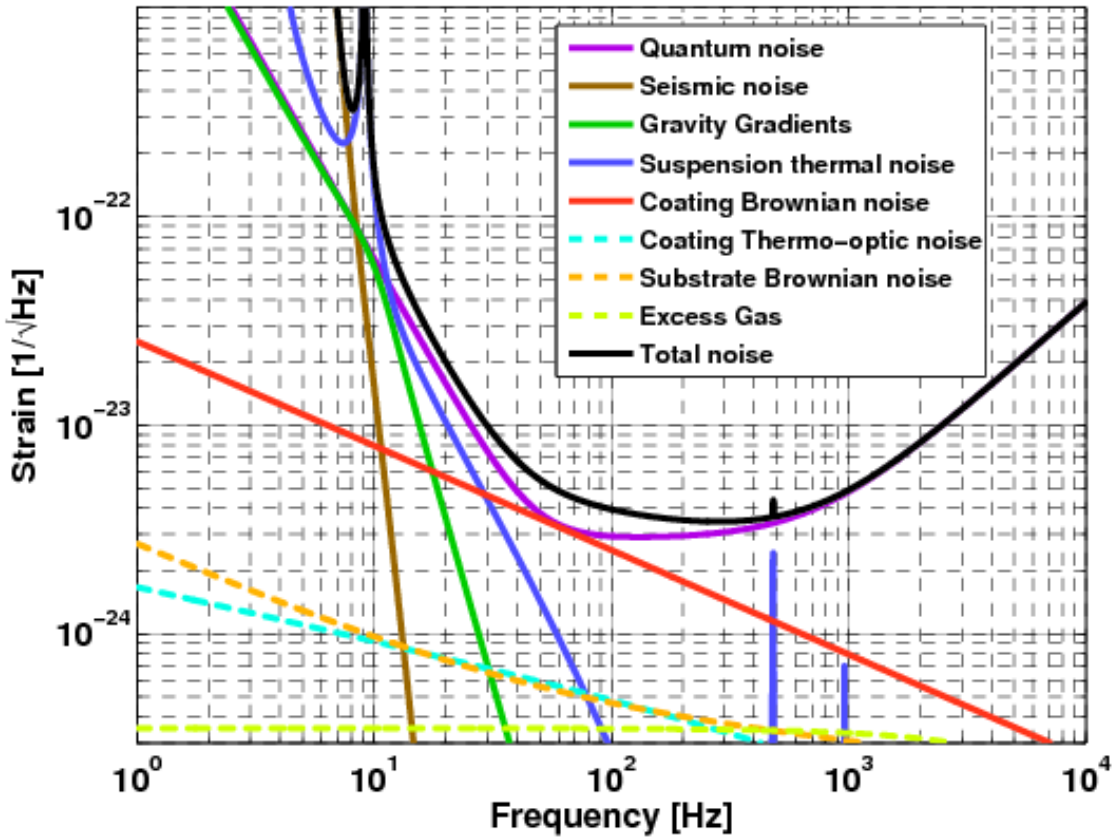



Image credit: S. Hild (2012)

# ML for noise cancellation in systems with non-linear coupling

**COMPLEX INSTRUMENT!**

Length Sensing and Control

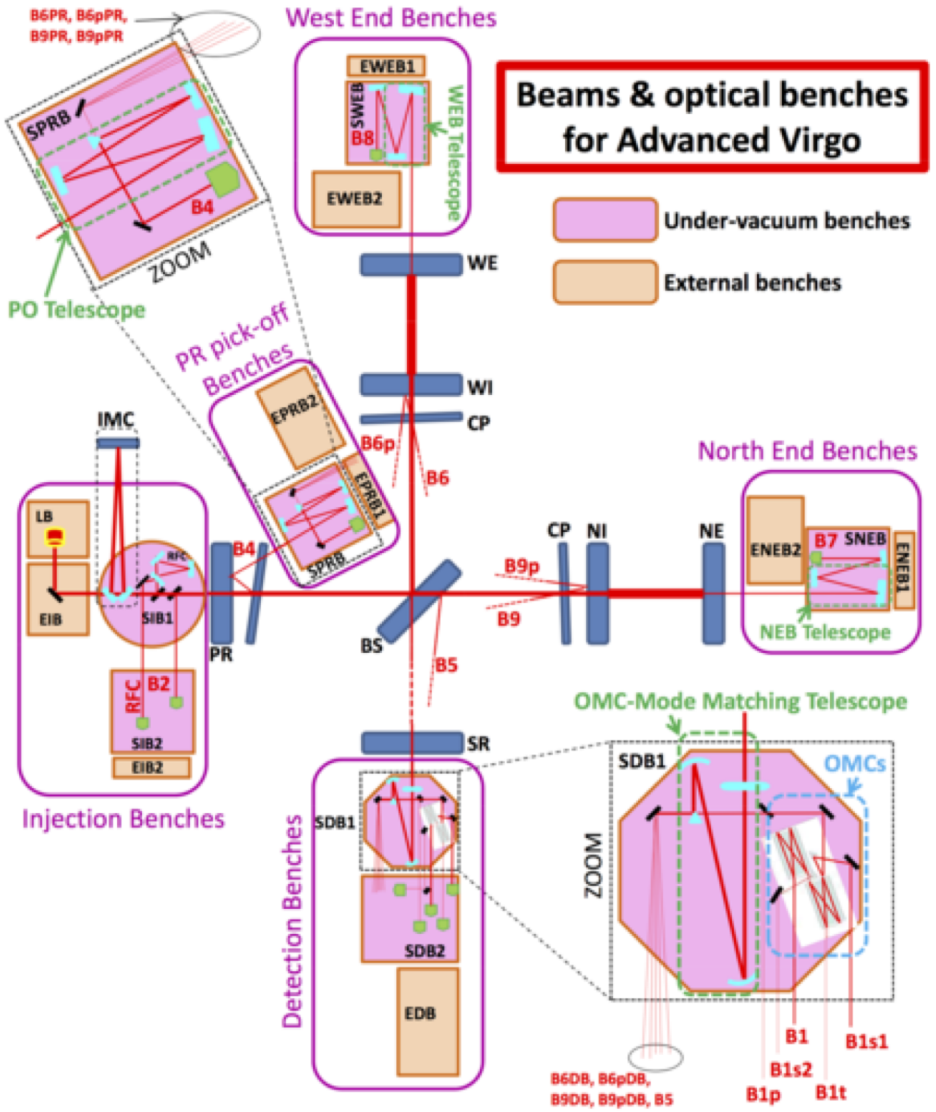
```
V1:LSC_NE_CORR 10000
V1:LSC_NI_CORR 10000
V1:LSC_PRCL 10000
V1:LSC_B8_56MHz_I 10000
V1:LSC_B8_56MHz_Q 10000
V1:LSC_B8_DC 10000
V1:LSC_DARM 10000
V1:LSC_DARM_56MHz_I 10000
V1:LSC_DARM_CORR 10000
V1:LSC_DARM_ERR 10000
```

Optical benches

```
V1:SDB2_LC_TX 10000
V1:SDB2_LC_TY 10000
V1:SDB2_LC_TZ 10000
V1:SDB2_LC_X 10000
V1:SDB2_LC_Y 10000
V1:SDB2_LC_Z 10000
```

Alignment Sensors

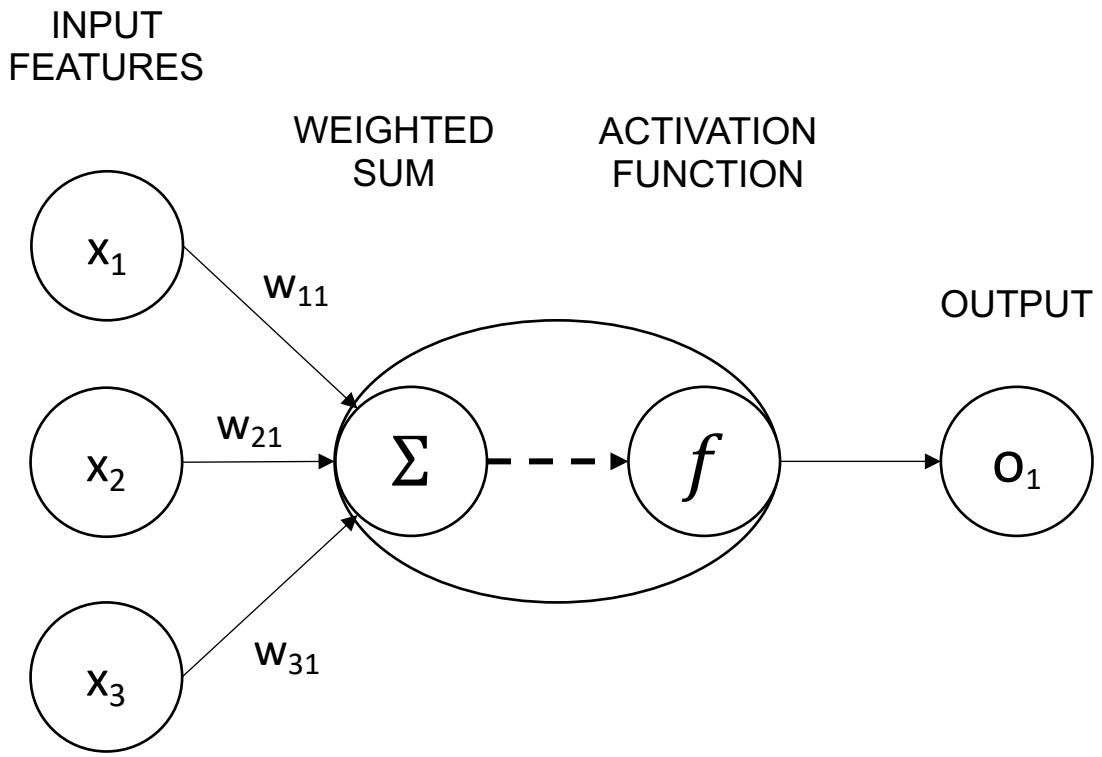
```
V1:ASC_B1p_QD1_H_56MHz_I 2000
V1:ASC_B1p_QD1_H_56MHz_Q 2000
V1:ASC_B1p_QD1_V_56MHz_I 2000
V1:ASC_B1p_QD1_V_56MHz_Q 2000
V1:ASC_B1p_QD2_H_56MHz_I 2000
V1:ASC_B1p_QD2_H_56MHz_Q 2000
V1:ASC_B1p_QD2_V_56MHz_I 2000
V1:ASC_B1p_QD2_V_56MHz_Q 2000
V1:ASC_B2_QD2_H_8MHz_I 2000
V1:ASC_B2_QD2_H_8MHz_Q 2000
V1:ASC_B2_QD2_V_8MHz_I 2000
V1:ASC_B2_QD2_V_8MHz_Q 2000
V1:ASC_B5_QD1_H_56MHz_I 2000
V1:ASC_B5_QD1_H_56MHz_Q 2000
V1:ASC_B5_QD1_V_56MHz_I 2000
V1:ASC_B5_QD1_V_56MHz_Q 2000
V1:ASC_B5_QD2_H_56MHz_I 2000
V1:ASC_B5_QD2_H_56MHz_Q 2000
V1:ASC_B5_QD2_V_56MHz_I 2000
V1:ASC_B5_QD2_V_56MHz_Q 2000
```



## Neural Networks (NNs)

- Universal Approximators
- Perform well on large datasets
- GPU implementation
- Computational load concentrated in training phase
- Already existing Deep Learning frameworks (Tensorflow, PyTorch, Keras…)

$$o_i = f\left(\sum_i w_i * x_i + b_i\right)$$



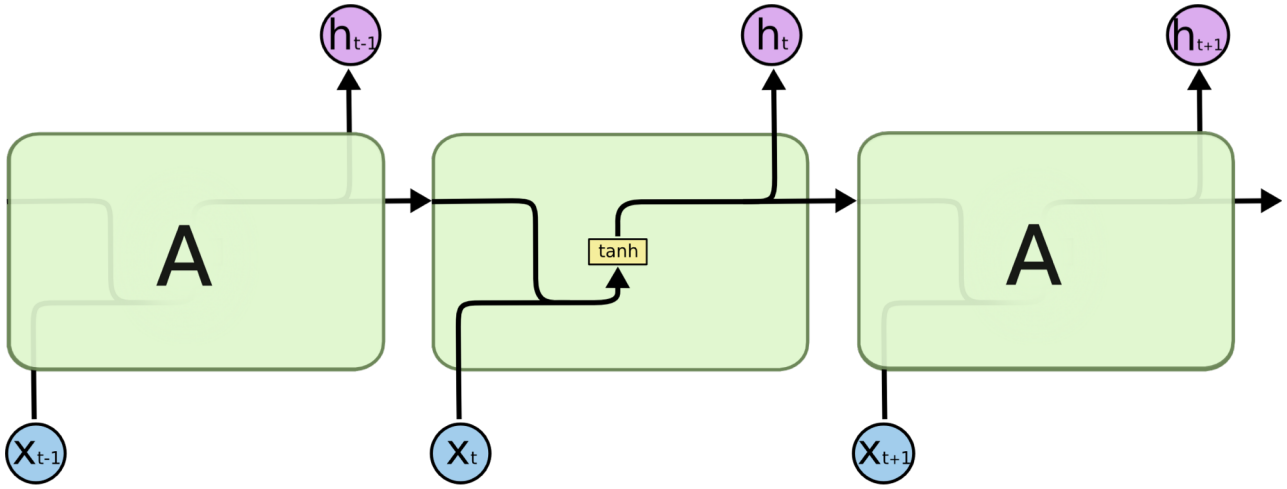
## Recurrent Neural Networks (RNNs)

**WHY?**

- Learn Time Dependencies

**HOW?**

- Pass on information from previous computation (feedback), which does not occur in regular neural networks




$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

!!!

**VANISHING GRADIENT PROBLEM**

Activation functions with values of gradients between 0 and 1: shallow layers learning decreases exponentially with number layers!

## Long Short-Term Memory (LSTM)

### Forget gate

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

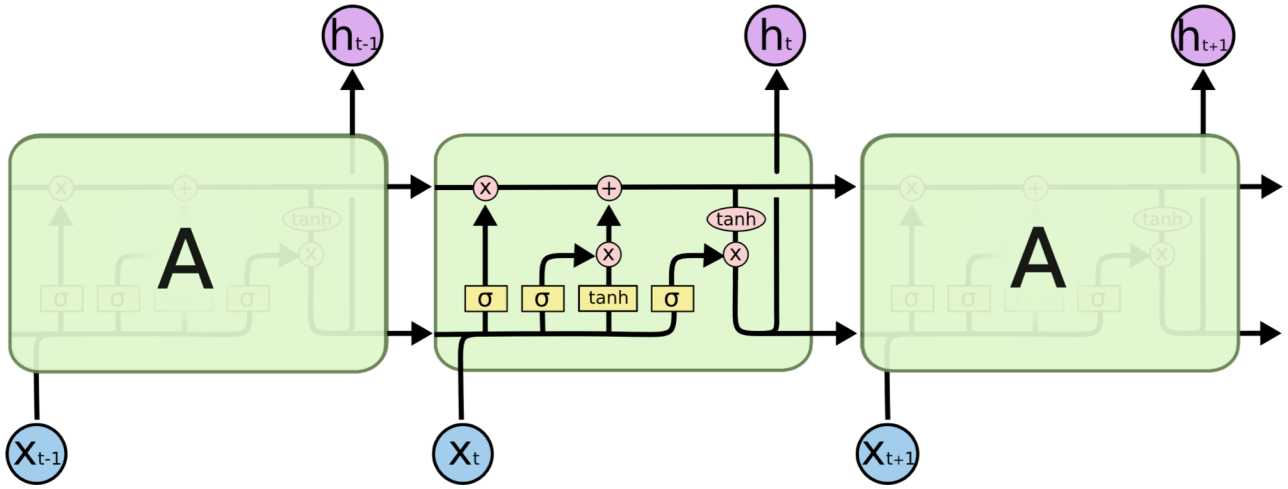
### Update gate

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

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

### Updated cell state

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




### Prediction

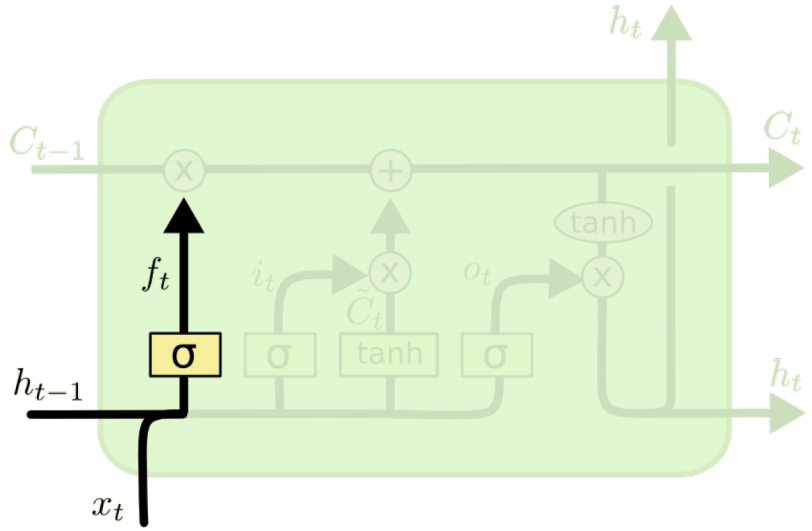
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

## Long Short-Term Memory (LSTM)

### Forget gate

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




### Update gate

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

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

### Updated cell state

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

### Prediction

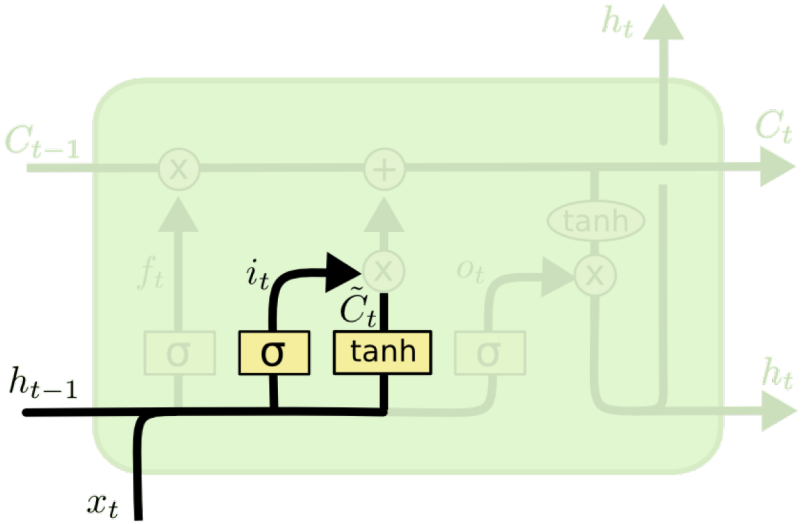
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

## Long Short-Term Memory (LSTM)

### Forget gate

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




### Update gate

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

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

### Updated cell state

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

### Prediction

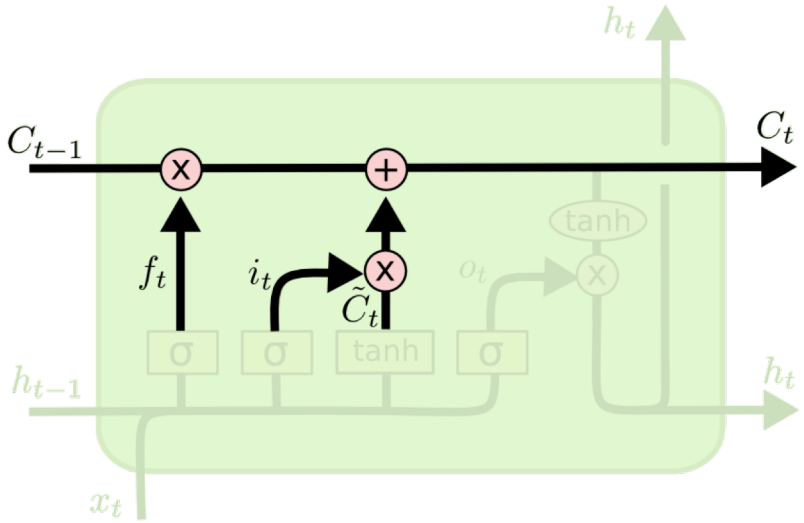
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

## Long Short-Term Memory (LSTM)

### Forget gate

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




### Update gate

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

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

### Updated cell state

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

### Prediction

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

$$h_t = o_t * \tanh\left(C_t\right)$$

## Long Short-Term Memory (LSTM)

### Forget gate

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

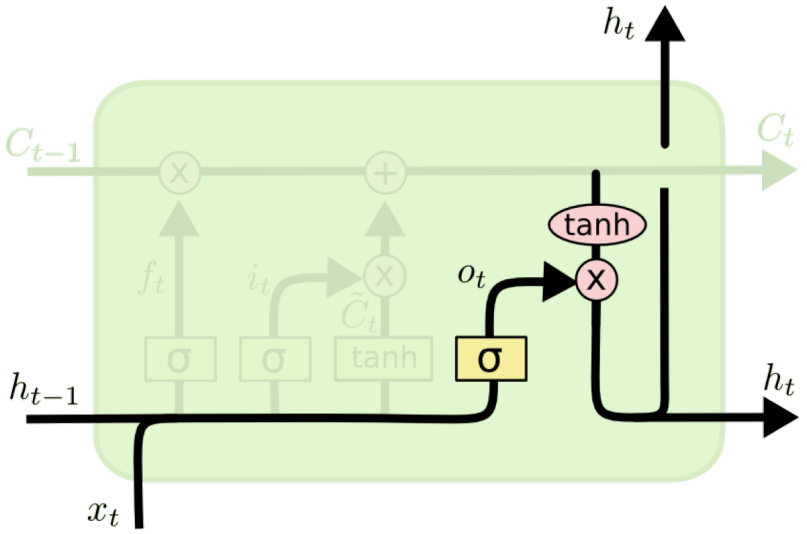
### Update gate

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

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

### Updated cell state

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



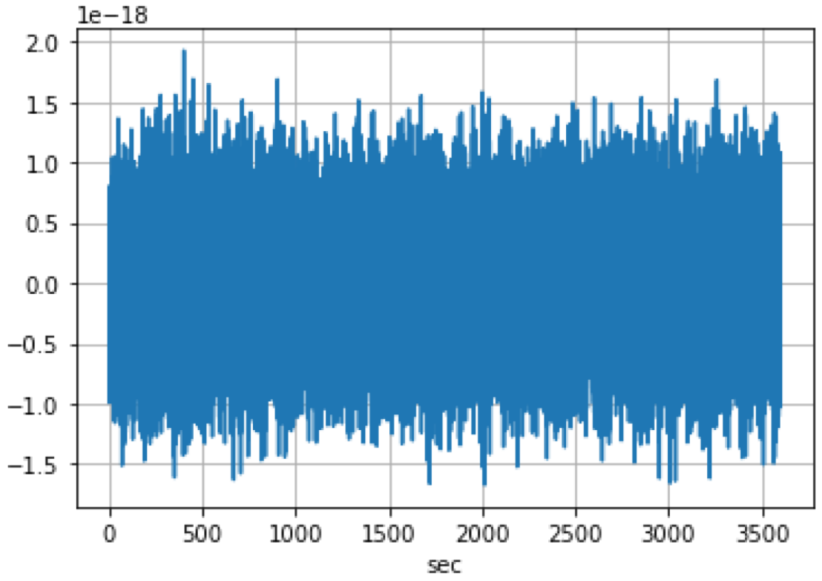

### Prediction

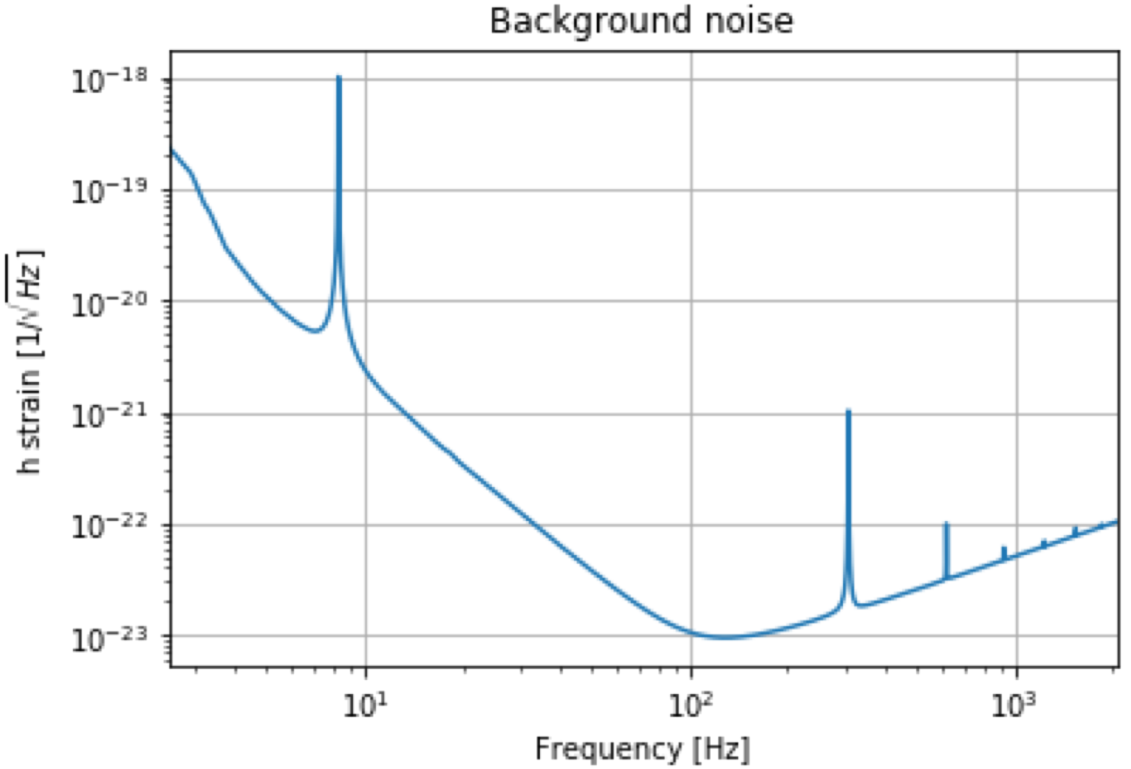
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

# LSTM FOR SIMULATED NOISE: A TOY MODEL

Generate a **background using** Advanced Virgo O2 design Sensitivity Curve (taken from calibration)



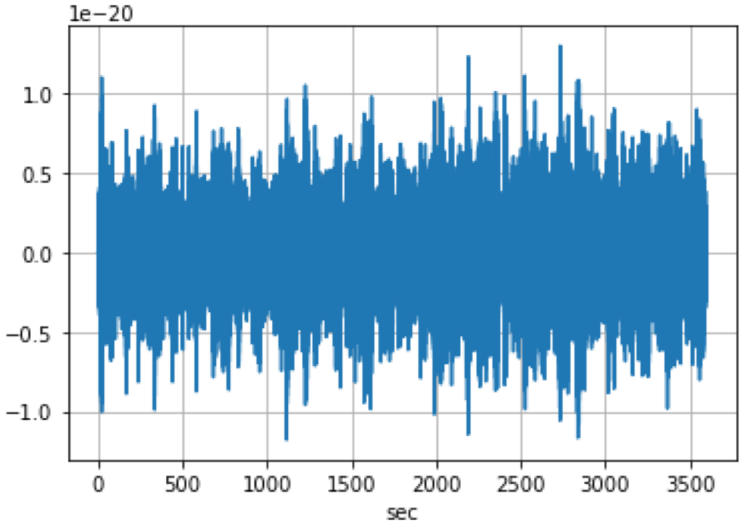

https://scientists.virgo-gw.eu/DataAnalysis/Calibration/Sensitivity/index.html

## LSTM FOR SIMULATED NOISE: A TOY MODEL

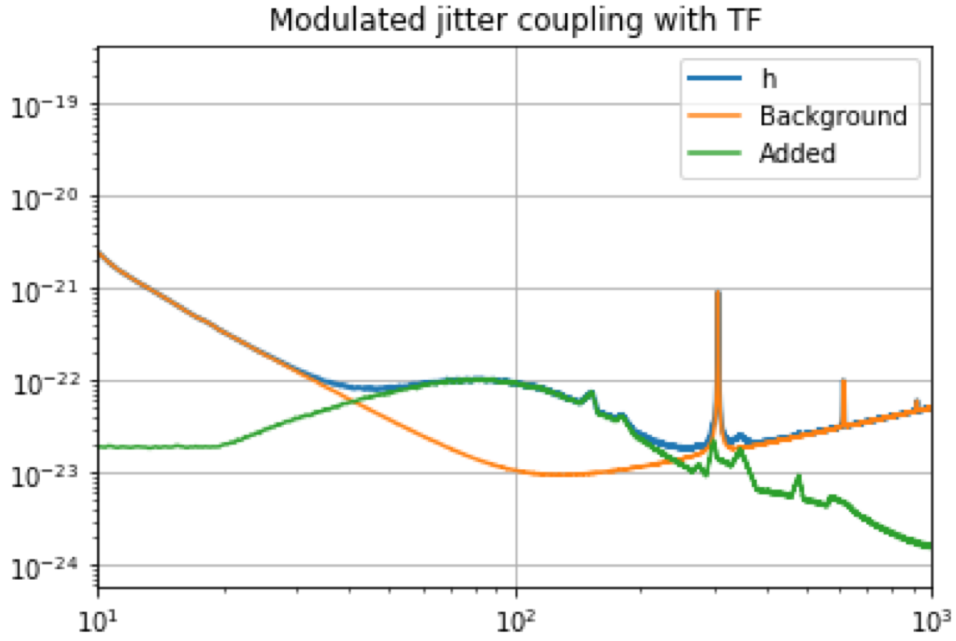
Add non-linearly coupled noise sources (e.g. **beam jitter** modulated by **seismic motion, see G. Vajente** [dn²] for more detailed examples):

$h_b + h_{nl}$ *Background + non-linear component*

$h_{nl} \propto f(wit1(t), wit2(t))$

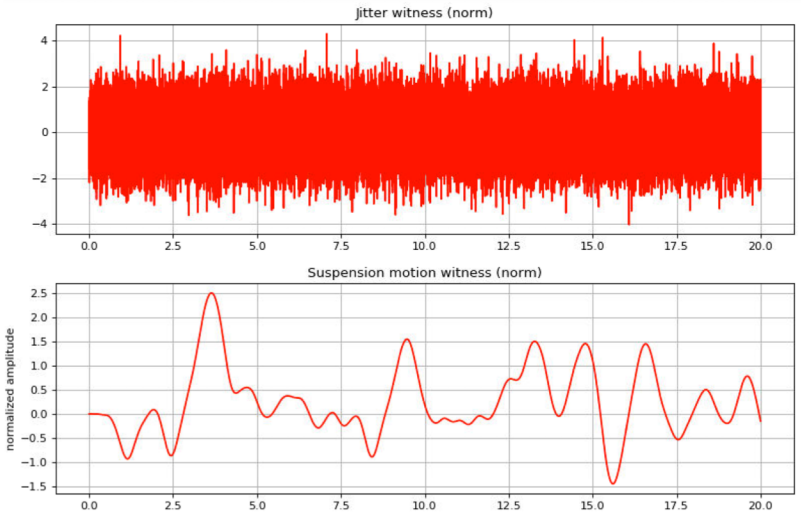


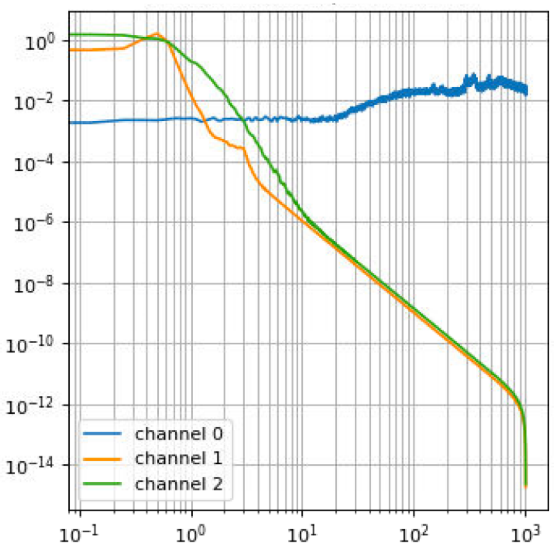

After the first Advanced LIGO observing run, lasting from September 2015 to January 2016, the two LIGO observatories in Hanford, Washington, and Livingston, Louisiana, underwent a series of upgrades [13]. The Handford observatory focused on increasing the amount of laser power circulating in the interferometer, which required using a high power oscillator [14]. The water required for cooling the laser rods flows through piping attached to the laser table, causing vibrations. This table also hosts an optical train for shaping the beam, impressing radio frequency sidebands, beam steering, and frequency stabilization. The water flow causes vibrations of these optical elements, which translates into beam jitter. Thermal fluctuations in the laser rods and mode mismatch in the optical resonators are also causing jitter variations of the the laser beam size—likely due to the turbulent water flow directly over the laser rods [15].



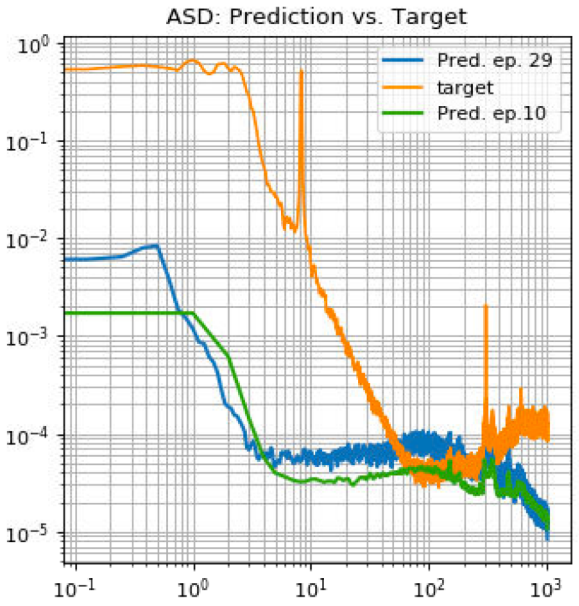
# ML for noise cancellation in systems with non-linear coupling

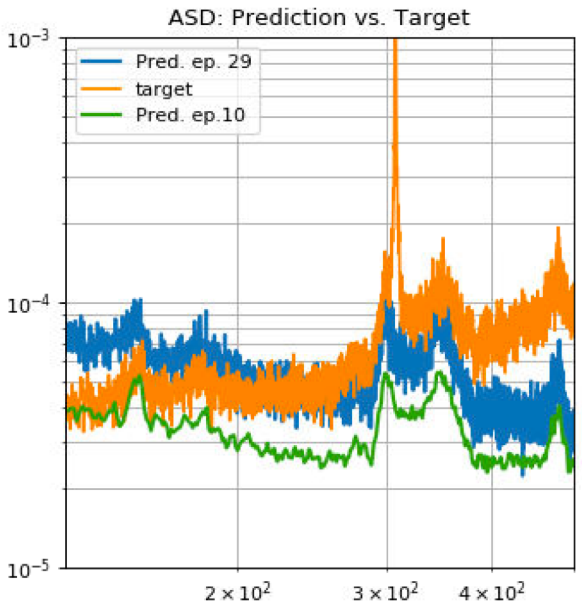
WITNESS CHANNELS




PREDICTIONS



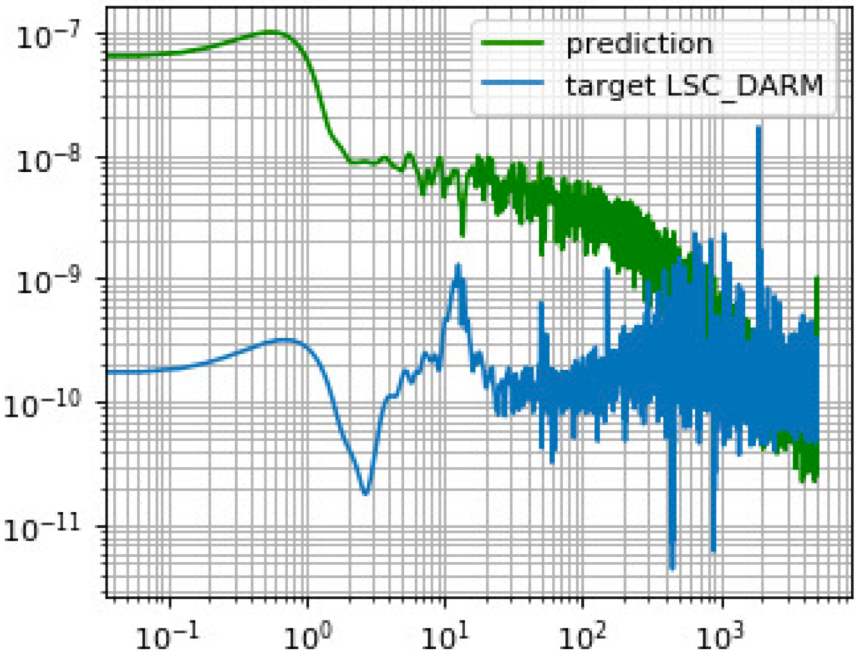

A. less

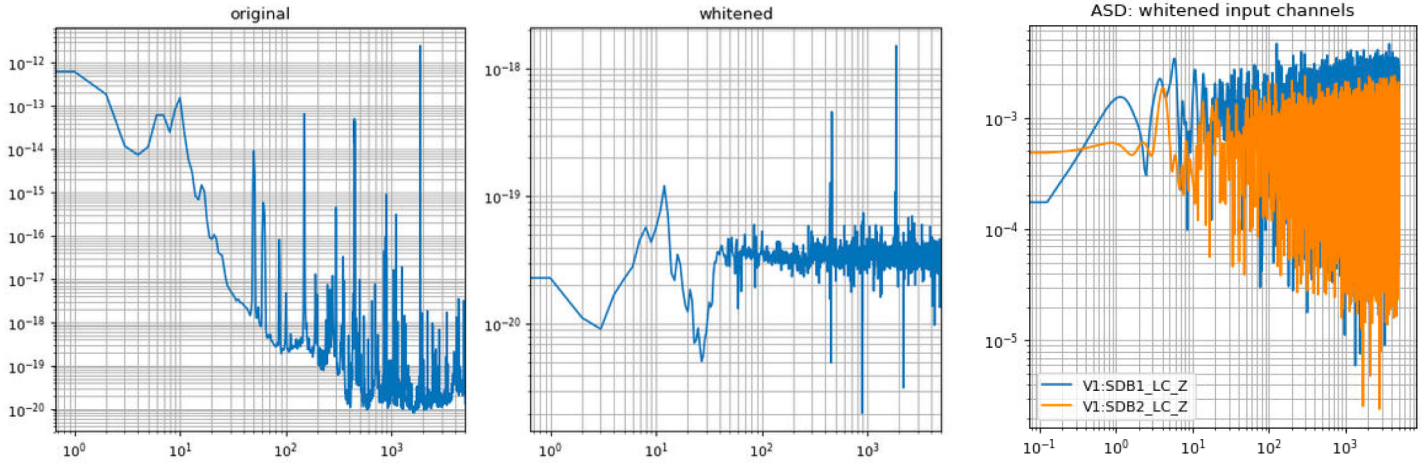
# ML for noise cancellation in systems with non-linear coupling

DETECTOR DATA

We try to learn the noisy target after whitening:

- Time Domain whitening
- Bad prediction with test cases (limited number!)
- Network learns mean value but not variance
- Whitening adds to computational time



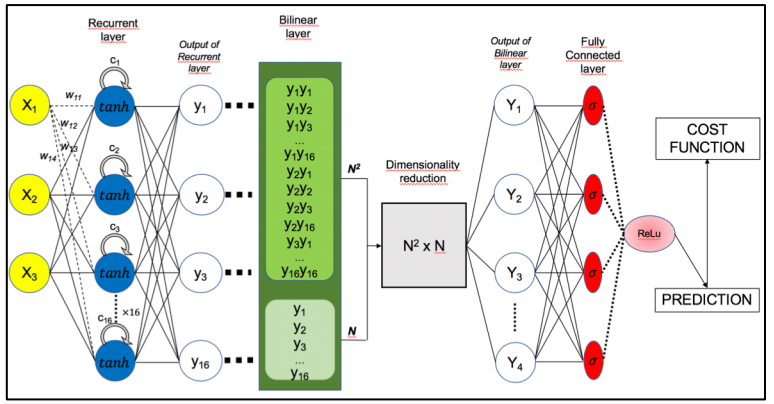




DIFFICULT TO TUNE RNN HYPERPARAMETERS!

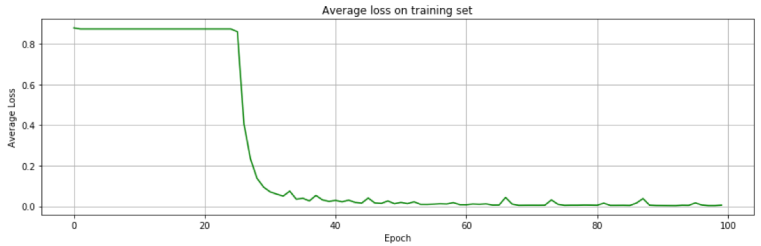
## Training the Network

Some fine tuning is needed to train LSTMs internal weights:

- Number and type of hidden layers
- Number of recurrent units?
- How to preprocess and filter data?
- How to scale input features
- Optimizer (ADAM, SGD, …
- Learning rate $\alpha = 10^{-1} \div 10^{-6}$ and its decay rate
- Weight initialization
- Cost function (MSE)
- Dropout and other regularization techniques
- Batch size
- Number of training samples




OPTIMIZATION PROBLEM!



## CONCLUSIONS

- LSTMs have been widely used for time-series forecasting (but are they good for regression?)
- RNNs can require long time to train
- Many Hyperparameters to tune
- Data Preprocessing is important
- How can ML be applied to interferometer data for noise subtraction?
- Can denoising ML algorithms run on data  offline or real time?