Draft version May 10, 2018
Typeset using LaTeX **twocolumn** style in AASTeX62

Extending the reach of gravitational-wave detectors with machine learning

Tri Nguyen

ABSTRACT

In this paper, I propose the use of Long Short-Term Memory (LSTM) neural networks, a subclass of machine learning algorithms, as a time series regression analysis technique to filter instrumental noises from gravitational-wave detectors at LIGO. Unlike traditional neural networks, LSTM networks can store and use information from their past inputs, thus making them robust in handling sequential data such as gravitational-wave signals. An LSTM network, once trained on the detector noise data from LIGO auxiliary channels, should be able to learn, forecast, and subtract both linear and non-linear noise coupling mechanisms. This would result in a sensitivity improvement, most greatly at the low-frequency limit where noise features are expected to be easier to learn, and allow the detection of gravitational-wave sources currently below the noise floor. I propose training and testing such network first on simulated detector data. Once the desired performance is reached (i.e. the network successfully removes the targeted noise sources without removing injected signals), I will apply the algorithm on real data from LIGO past observations. I will be able to uncover more physical properties of binary sources, especially during their inspiral stage, thanks to the sensitivity improvement in the low-frequency limit.

1. INTRODUCTION

1.1. *Noise Regression Analysis at LIGO*

The Laser Interferometer Gravitational-Wave Observatory (LIGO) is the world's largest gravitational-wave observatory. Adopting a Michelson interferometer design, LIGO detects passing gravitational waves by measuring the induced differential arm length (DARM) between its two perpendicular 4-km arms (Adhikari 2004; Tiwari et al. 2015; LIGO Scientific Collaboration 2007). Since its sensitivity upgrade in 2015 (The LIGO Scientific Collaboration 2015), LIGO has observed gravitational waves from multiple stellar-mass black-hole and neutron star mergers (Abbott, B. P. et al 2016). Many of these objects, however, are still lying below the detector sensitivity limit. LIGO's noise sources include contributions from instrumental and environmental noises, each coupling to the DARM with a different, both linear and non-linear, mechanism. A detailed study of these mechanisms would allow us to filter out the noises, improve LIGO's sensitivity, and make gravitational-wave detections a more routine occurrence.

The current noise regression method at LIGO is based on the Wiener-Kolmogorov filter, which minimizes the squared error between the gravitational-wave channel and the predicted noises from physical environment monitor (PEM) channels (Tiwari et al. 2015; LIGO Scientific Collaboration 2007). The Wiener filter, however, fails to remove non-linear contributions. In fact, characterizing and filtering out these non-linear noise sources has proven to be a challenging process, because the coupling mechanisms are often sophisticated. For example, two or more noise sources may interfere before coupling to the DARM.

1.2. *Neural Networks as a Noise-Filtering Technique*

Despite the complexity of the non-linear noise coupling mechanisms, a neural network may be able to learn them. A neural network is a non-parametric, supervised machine learning algorithm often used for data classification and clustering. Modeled loosely after a human brain, a typical network may consist up to thousands of nodes, each carrying parameters characterizing the features of the data. The network learns by looping over a labeled dataset (called the training set) and minimizing a set loss function via gradient descent. Once sufficiently trained, neural networks are capable of recognizing highly complex patterns, such as language models, stock market, etc.

In this project, I propose to filter LIGO's noise sources by using Long Short-Term Memory (LSTM) networks, a subclass of neural networks which specializes in processing sequential input, like the time-series from LIGO's data and PEM channels. A major strength of LSTM networks over traditional neural networks is their ability to store and use information from previous inputs. As a result, LSTM networks are capable of taking into account the long-term dependencies in the data (Olah 2015). They are frequently used in speech recognition, grammar learning, and even DNA pattern recognition (Karpathy 2015). Given enough training data, an LSTM network should be able to learn, predict, and subtract

both linear and non-linear noise coupling mechanisms, resulting in an improvement in LIGO's sensitivity.

2. OBJECTIVES

2.1. *Fundamental vs. Non-fundamental Noises*

LIGO's noise sources can be categorized into fundamental and non-fundamental noise sources (The LIGO Scientific Collaboration 2015). Fundamental noises are those imposed by quantum and statistical mechanics; these include photo shot noise, thermal noise, etc. Because fundamental noises define LIGO's baseline sensitivity limit, they can only be reduced by improving the detector design. On the other hand, non-fundamental noises consist of instrumental and environmental effects, such as seismic noise, magnetic noise, beam jitter, etc. Non-fundamental noises can be removed, given there exists witness channels monitoring the disturbances.

The goal of the network is to successfully predict and filter out the non-fundamental noises, while keeping the gravitational-wave signals and fundamental noises intact.

2.2. *Criteria for Success*

Given a witness channel i at some time t, the network will predict the induced DARM at some time $t' > t$. Mathematically speaking,

$$\tilde{D}(t') = f[w_i(t)], \quad t' > t$$

where $f[w_i(t)]$ is some non-linear function learned by the network. In general, we do not know the full functional form of $f[w_i(t)]$. The performance of the network is instead evaluated by computing the coherence $c(f)$ between the DARM $D(t')$ and the clean DARM $D_{clean}(t') = D(t') - \tilde{D}(t')$. The coherence measures the correlation between these two DARM channels at each frequency f and can be computed from their Fourier transforms. It is related to the ideal noise suppression factor $r(f)$ by the relation:

$$r(f) = \frac{1}{\sqrt{1 - c^2(f)}}, \quad 0 \leq c \leq 1$$

For example, if the coherence between $D(t')$ and $D_{clean}(t')$ at some frequency is 0.9, the noise reduction factor at that frequency will be approximately 2.3 (M Coughlin 2014). The goal of this project is to achieve a noise reduction factor of 2, corresponding to a coherence of about 0.87.

2.3. *Expected Outcomes*

Once the network reaches the desired performance, it will be ready to run on real-time gravitational-wave data. I expect an overall sensitivity improvement, leading to an increase in the number of gravitational-wave sources detected. Moreover, as low-frequency noise features are often stronger (louder bins in the power spectral density) and thus should be easier to learn, I predict an additional increase in sensitivity at the low-frequency limit. This improvement is especially important for observing binary sources because it increases the observation time frame during the inspiral stage where the orbital frequency is generally low.

3. APPROACH AND WORK PLAN

I propose the following analysis and timeline for developing the computational procedure: **1.** (week 1) Generating mock data. **2.** (week 2-6) Building the optimal LSTM network. **3.** (week 7-10) Running on real LIGO data.

3.1. *Generating Mock Data*

In this step, I will generate mock data to train and test the LSTM network. To introduce non-linearities, bilinear data are generated by adding the product of angular sensing noise (from ASC channels) and beam spot motion (from beam spot channels) to the DARM. Note while the true coupling mechanisms are not known (as the objective of this project is to characterize these mechanisms), I expect the mock data to be similar enough to real data and sufficiently capture their complexity. To ensure the network only removes the targeted noises, I will also inject gravitational-wave signals and other noise sources into the mock, and later check if they are removed or distorted.

Generating mock data should be straight-forward, so I expect this step to consume the least amount of time and propose a timescale of less than one week.

3.2. *Building the Optimal LSTM Network*

In this step, I will focus on building the optimal LSTM network via hyperparameter tuning. In machine learning, hyperparameters are those that govern the learning process, such as the learning rate or the number of hidden layers and hidden units. They are set before training, and can heavily affect the optimization and performance of the neural network. Different algorithms usually require different sets of hyperparameters. Finding the optimal set of hyperparameters, or hyperparameter tunning, is an empirical process. In other words, it requires experimenting with the network on different sets of hyperparameters and evaluate their performance. Therefore, I expect this step to be the most challenging, time-consuming, and computationally expensive. I propose a timescale of about 5-6 weeks.

3.2.1. *Hyperparameter Tunning*

I will apply the holdout cross-validation method. In particular, the mock data are partitioned into a training set and a test/development set. After each epoch, or each iteration through the training set, the network is run on the test set. As mentioned in Section, the performance is evaluated by computing the coherence between DARM and the clean DARM. This gives an insight on how the network performs on an independent, unknown dataset. There are three possible outcomes:

- High Bias, High Variance: The network performs poorly on both the training set and the test set. This is due to underfitting. An immediate diagnosis is to check if the gradient descent converges and subsequently increase the number of iterations and/or the learning rate. If underfitting persists, I propose to increase the network's size (e.g. add more hidden layers and hidden units) as this will allow the network to recognize more complex patterns (more non-linearities).

- Low Bias, High Variance: The network performs well on the training set but poorly on the test set. This is due to overfitting and is one of the biggest problems in machine learning. The general solution is to apply regularization during the training process, or, if regularization is already used, increase the degree of regularization. Regularization reduces overfitting because it, roughly speaking, introduces noises into the training data. In addition, generating more training data also helps reduce overfitting.

- Low Bias, Low Variance: The network performs well on both the training set and the test set. This is the ideal result. The hyperparameters are optimal for the analysis.

Alternatively, changing the network's architecture, such as the network's type or the output function in each layers, may also reduce underfitting and overfitting. This serves only as a last resort because it requires rebuilding the analysis pipeline.

Finally, another important set of hyperparameters is the size of the training set and test set. In principle, each set should be large enough to represent the mock data. Because more mock data can be generated as needed, the training set and test set are usually the same size. In practice, however, this ratio may vary slightly.

3.2.2. *Current Analysis Pipeline*

The current analysis pipeline, DeepClean, is available in the Python programming language. It consists of three separate LSTM networks: Low, Mid, and High. Each uses a different set of hyperparameters and optimization (gradient descent) algorithm, and runs on a different frequency band. The frequency cutoffs are 3-19 Hz, 18-40 Hz, and 37-80 Hz respectively. On each run, DeepClean loops over each frequency band and performs subtraction using the assigned network; the final subtraction will be a combined result of three networks. Each network is trained and tested independently.

I will use DeepClean as my main analysis pipeline and make some adjustments if required. For example, the number of frequency bands, itself a hyperparameter, may be subject to change.

3.3. *Running on real LIGO data*

Once the optimal set of hyperparameters is chosen, I will test DeepClean on real data from LIGO's past observation. If DeepClean successfully removes non-fundamental (subtractable) noises without removing the signals or the fundamental (non-subtractable) noises, it will be ready to run on real-time, future LIGO's data.

Because the training and testing processes are similar for real and mock data, I expect this step to be relatively straight-forward, possibly with some minor twitches of the hyperparameters. I propose a timescale of about 3-4 weeks.

REFERENCES

Abbott, B. P. et al. 2016, Phys. Rev. Lett., 116, 061102

Adhikari, R. 2004, PhD thesis, Massachusetts Institute of Technology, Massachusetts, USA

Karpathy, A. 2015, The Unreasonable Effectiveness of Recurrent Neural Networks

LIGO Scientific Collaboration. 2007, arXiv, 0711.3041

M Coughlin, J Harms, e. a. 2014, Classical and Quantum Gravity, 31, 215003

Olah, C. 2015, Understanding LSTM Networks

The LIGO Scientific Collaboration. 2015, Classical and Quantum Gravity, 32, 074001

Tiwari, V., et al. 2015, Class. Quant. Grav., 32, 165014