Gravitational Wave Detection and Parameter Estimation Using Deep Neural Networks

Rui Lan Jiaxi Nie Dongwei Shi
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{ruilan2, nie9, dshi9}@illinois.edu

Abstract

Gravitational wave (GW), predicted by Albert Einstein in 1916 based on his General Theory of Relativity, was first detected in 2015 by the Laser Interferometer Gravitational-Wave Observatory (LIGO) [1]. This year, the leading scientists in the Royal Swedish Academy of Sciences win the Nobel Prize in physics based on their contributions to GW detection of merging black holes. These all make GW a promising and exciting research area. In this report, we show how deep neural networks can be applied to GW detection and parameter estimation. We also introduce our multi-GPU training approach which can significantly reduce the training time of deep neural networks while preserving the accuracy as the original code at the testing time.

1. Introduction

Recently, the astrophysicists discovered a new kind of GW event caused by the merging of two neutron stars and the 2017 Nobel Prize in physics was awarded to the scientists that founded LIGO, which detected a GW event for the first time in 2015, which was caused by the merging of two black holes. The detection of GW verified the hypothesis of the Einsteins General Relativity and marked a significant milestone in the exploration of astronomy physics in humans history. It is apparent that the GW physics is an exciting area to research on.

It is essential to have a fast and accurate algorithm to detect GW from other time-series signals observed. Due to the error of measurement and the nature of GW, the observed GW signals tend to be highly noisy. The matched filtering approach, which involves convolutions of input GW signal and the template waveform, is used for searching GW by LIGO [2]. This method takes the newly discovered GW signal as the input and applies convolution operations on that signal with multiple templates. The matched filtering method works well for these years' GW detection. However, there are certain drawbacks with this approach, one of which is the intensive computational complexity of the convolutions because the size of the template bank used by LIGO is over 300,000 [2]. Because of the large size of templates, it takes an undesirably long computational time at the testing stage which prohibits researchers to detect the GW in real-time by using matched filtering method.

To enable the real-time GW detection, George and Huerta [3] first proposed a method to use deep neural networks. The new deep learning approach can not only detect GW with high accuracy, but also perform parameter estimation to extract the information of a certain GW which would aid the astrophysicists further understand the universe. Specifically, they succeeded in estimating the masses of the two merging black holes which is a two-parameter regression problem [3]. To do these, they solve the detection and parameter estimation into two different neural networks, namely, classifier and predictor. The classifier and predictor both contain three convolutional layers and two fully connected layers. The only difference between those two networks is the last layer which makes those two networks perform classification and regression problem respectively. They showed the deep neural networks approach can obtain a much shorter inference time at the testing stage while preserving the accuracy at the same level with the results generated by LIGOs matched filtering method [3].

Although George and Huerta's deep learning method [3] can handle the problem quite well, there are several overhead to examine, such as the training time for these deep networks are still long and the fact that the predictor in their paper can only generate value of two masses while ignoring other essential parameters. In this report, we introduce our classifier and predictor which can detect and extract parameters from highly noisy GW signal containing masses and spins of both two black holes or neutron stars. information. We also show that our multi-GPU training approach can significantly reduce the training time of neural networks by half.

The data we use is generated scientific simulations by

NCSA Gravitation Group, as the training our networks requires a significant amount of data and it is hard to reach the requirement by using existing real LIGO data. The data we use has both signal and label. The signal part is a vector contains 8192 points which represent the information of a one-second-long GW. We show an example GW signal randomly selected in our training dataset in Figure 1. The label contains four parameters, which are the masses and the spins of two merging black holes. The dataset we use includes 81,900 training signal and 9,100 testing signal. We use the signal differently in classification and regression problems. The detailed methods will be discussed in the following section.

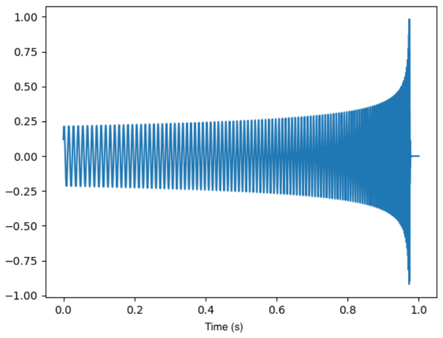


Figure 1. Example of a GW randomly selected from our dataset.

2. Methods

2.1. Detection

As we mentioned before, the GW is hard to detect because of its highly noisy nature. We train our classifier to detect the incoming signal. The input contains two kinds of signals, i.e., pure Gaussian noise and GW injected with Gaussian noise. To simulate this, we add Gaussian noise into our dataset. We use signal-to-noise ratio (SNR) to represent the relative relationship of amplitudes of GW and noise. When the SNR is smaller than 1, the Gaussian noise dominates in the signal. In Figure 2, the top signal in blue represents the GW injected with Gaussian noise with SNR equals to 0.5 and the signal in red shows the pure GW. The bottom signal represents the pure Gaussian noise.

We use a neural network with four-convolutional-layer in this project to perform the inference task. The idea of this network comes from George and Huertas paper [4], and Figure 3 shows the overall structure of this network. For the convolution layers, we use filter size as [16, 32, 32, 64] for the four convolutional layers. As a classification problem, we add a softmax layer at the end of our classifier to generate the one-hot label. Specifically, [0, 1] indicates the

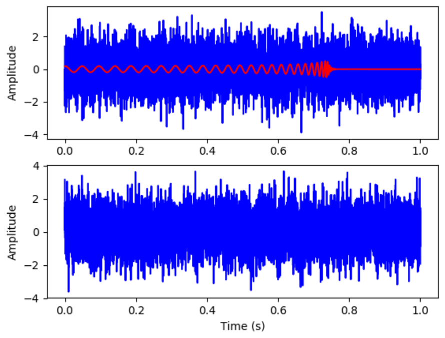


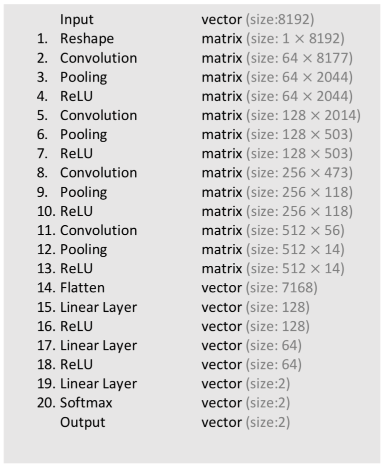
Figure 2. Example of pure Gaussian noise and GW injected with Gaussian noise. Top (blue): GW injected with Gaussian noise with SNR equals to 0.5. Top (red): pure GW. Bottom: pure Gaussian noise.

noisy signal contains GW and [1, 0] indicates pure Gaussian noise. We also experiment with other Convolutional Neural Networks (CNN) with only 3-convolutional layers. It turns out the 4-convolutional-layer network outperforms for our detector. We experiment on various hyperparameter settings to determine the best model parameters. The result shows that learning rate of 0.001 works better for our classifier. Suggested by George [3], in order to capture the GW with different time offsets, we randomly shift our GW by a quarter of the signal length to simulate the real-life situation, in that real-time detection is not performed continuously but periodically in a fine interval. The input GW may not contain the full one-second information. As the SNR of the input GW signal may vary in a wide range, we have to ensure our classifier can work with different SNR during inference. Specifically, we inject all the GW with Gaussian noise with SNR equals to 0.25 during the training, and this method guarantees that our model can detect GW accurately when testing. We adopt Stochastic Gradient Descent (SGD) optimizer and batch size of 128.

2.2. Parameter Estimation

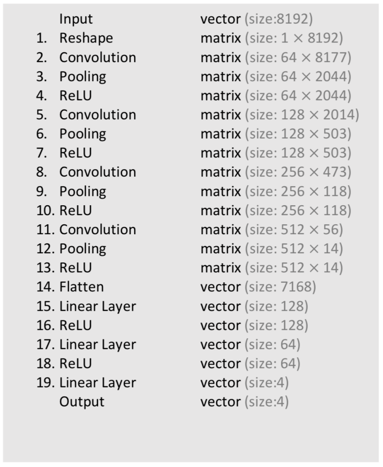
In addition to the classifier, we also construct another deep neural network to extract the information of the detected GW, which includes masses and spins of two merging black holes. This predictor is a 4-parameter regression problem which requires deeper CNN and more powerful training strategy.

The input of our predictor is the highly noisy GW detected by our classifier. In our project, we feed the predictor with the signal similar to the top one (blue) in Figure 2. The four parameters we want to predict, i.e., two masses and two spins, corresponds with the signal. The CNN architecture is shown in Figure 4. We adopt similar structure as our classifier with larger filter sizes, i.e. [32, 64, 128, 64]



	Input	vector (size:8192)
1.	Reshape	matrix (size: 1 × 8192)
2.	Convolution	matrix (size: 64 × 8177)
3.	Pooling	matrix (size: 64 × 2044)
4.	ReLU	matrix (size: 64 × 2044)
5.	Convolution	matrix (size: 128 × 2014)
6.	Pooling	matrix (size: 128 × 503)
7.	ReLU	matrix (size: 128 × 503)
8.	Convolution	matrix (size: 256 × 473)
9.	Pooling	matrix (size: 256 × 118)
10.	ReLU	matrix (size: 256 × 118)
11.	Convolution	matrix (size: 512 × 56)
12.	Pooling	matrix (size: 512 × 14)
13.	ReLU	matrix (size: 512 × 14)
14.	Flatten	vector (size: 7168)
15.	Linear Layer	vector (size: 128)
16.	ReLU	vector (size: 128)
17.	Linear Layer	vector (size: 64)
18.	ReLU	vector (size: 64)
19.	Linear Layer	vector (size:2)
20.	Softmax	vector (size:2)
	Output	vector (size:2)

Figure 3. Network structure of the classifier.



	Input	vector (size:8192)
1.	Reshape	matrix (size: 1 × 8192)
2.	Convolution	matrix (size: 64 × 8177)
3.	Pooling	matrix (size: 64 × 2044)
4.	ReLU	matrix (size: 64 × 2044)
5.	Convolution	matrix (size: 128 × 2014)
6.	Pooling	matrix (size: 128 × 503)
7.	ReLU	matrix (size: 128 × 503)
8.	Convolution	matrix (size: 256 × 473)
9.	Pooling	matrix (size: 256 × 118)
10.	ReLU	matrix (size: 256 × 118)
11.	Convolution	matrix (size: 512 × 56)
12.	Pooling	matrix (size: 512 × 14)
13.	ReLU	matrix (size: 512 × 14)
14.	Flatten	vector (size: 7168)
15.	Linear Layer	vector (size: 128)
16.	ReLU	vector (size: 128)
17.	Linear Layer	vector (size: 64)
18.	ReLU	vector (size: 64)
19.	Linear Layer	vector (size:4)
	Output	vector (size:4)

Figure 4. Network structure of the predictor.

to extract more information from our input signals. Also, instead of a softmax layer at the final stage, we use linear layer to produce MSEs because of the nature of the regression problem (the output should be in continuous space). As we mentioned before, we expect our predictor can work for different SNR with training it only once. However, the regression problem is typically harder to converge than the classification problem, and we find the approach with fix SNR training in the classifier cannot work for the predictor. Thus, we construct a function to decrease upper bound of SNR from 100 to 3 as a function of training step and fix the lower bound at 0.2. We randomly choose SNR from the range between 0 and the upper bound SNR at each training step. We show this method is useful to help our predictor remember the signal at a wide range of SNR so that it performs equally well in testing process. To compensate the difficulty of convergence in this model, we decrease the learning rate to 0.0001 as higher learning rate such as 0.001 cannot lead us to the global minima. We use SGD optimizer during training and batch size of 128.

2.3. Multi-GPU Training

The deep learning approach significantly reduce time complexity of the inference process compared with traditional matched filtering approach used by LIGO [3]. However, considering the large amount of training data, the time complexity becomes a new problem. Hence, to accelerate the training process, we implement a multi-GPU training approach as well.

Each compute node of the Illinois Campus Cluster equips with two NVIDIA Tesla K40 GPUs. The deep learning framework we apply in this project, the TensorFlow, can harness this environment to run multiple training process concurrently [5].

The most common way to training a deep neural network with multiple GPUs is dividing the data into multiple subsets (data approach). Each GPU inside the compute node is running an identical deep neural network model replica and correspondingly training each subset data. By waiting all the GPUs finished its work, the multiple GPUs training framework updates the model parameters concurrently. Note that each GPU inside the compute node should have similar compute power and memory to run an entire deep neural network. We applied two NVIDIA Tesla K40 GPUs in our training process.

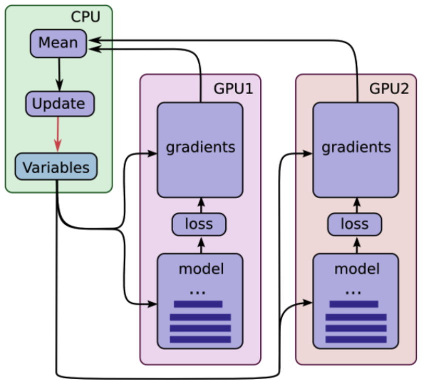


Figure 5. The data approach multi-GPU training model [6].

The diagram of our multi-GPU model is shown as Figure 5 by Tensorflow [6]. As the diagram shows, we first distribute the same amount of data into two identical GPUs with two sets of trainable variables. After the two GPUs finish their computation, the CPU compute the average gradient and update the parameters using the average gradient. By using this data-parallel method, the GW detection training process demonstrates a higher compution efficiency while keeping the same accuracy.

3. Results and Discussion

In the following sections, we show our results of the classifier, predictor, as well as the multi-GPU training. We also discuss our results and make comments on possible improvements.

3.1. Detection

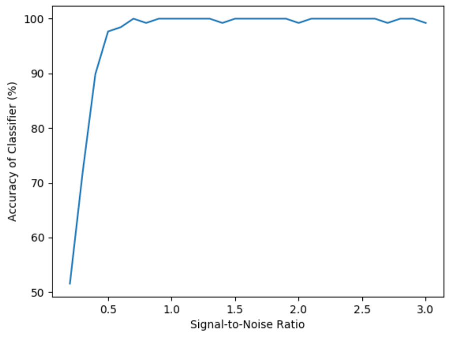


Figure 6. SNR-accuracy plot of the classifier.

Figure 6 shows the SNR-accuracy plot for our classifier. We show our classifier can achieve nearly 100% accuracy when the SNR is greater than 0.5, and this classifier can be applied to encourage GW research for detecting new GW events. As discussed in George and Huertas paper [3], the deep learning approach provides people with a method to detect GW efficiently and enable the real-time multi-messenger astrophysics. The result of different learning rates with accuracy is attached in Figure 7, as we recognize the importance of hyperparameters affecting the deep learning model. We observe the trade-off between coverage rate and accuracy. Specifically, high learning rate may not lead us to the local minimum and low learning rate makes the model difficult to converge. In the classifier of this project, a learning rate of 0.001 performs well.

3.2. Parameter Estimation

We demonstrate the predictor result in Figure 8. The top plot shows the mean-squared-error (MSE) to SNR and the bottom plot shows the relative error to SNR. We show that by using our decreasing SNR training approach, the relative error is less than approximately 40% of the predicted parameters and the ground truth. However, there are several possible improvements based on our method. Although our predictor shows a relatively low MSE during the testing, the relative error can be further improved. Generally, we think using deep neural networks with good hyperparameter selection and training methods can solve this problem, and by

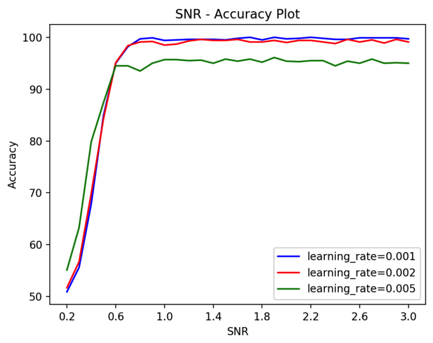


Figure 7. SNR-accuracy plot of the classifier with different learning rate.

decreasing the lowest SNR, the model can be applied on by trying new architecture as well.

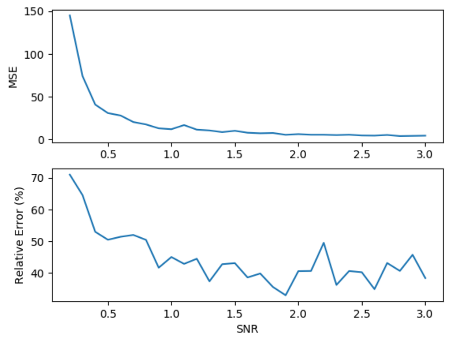


Figure 8. SNR-MSE and SNR-relative error plot of the predictor.

3.3. Multi-GPU Training

The multi-GPU training result is shown in Figure 9. Due to the limit of computational resources we can only test our model on 2 Tesla K40 GPUs. It is apparent that the multi-GPU code can reduce the training time by more than one third comparing with only one GPU from the left plot of Figure 9. The right plot shows our multi-GPU code can keep the same accuracy with the original one-GPU approach. We apply our multi-GPU code to both classifier and predictor. The results here is obtain from our classifier.

For more GPUs, we believe our approach cannot work as good as 2-GPUs because the expensive synchronization is required between GPUs. We reckon this multi-GPU model is a good start point for the distributed GPU method. As suggested by Goyal *et al*. [7], by increasing the batch size and adjusting the learning rate, the synchronization time of different GPUs can be reduced significantly.

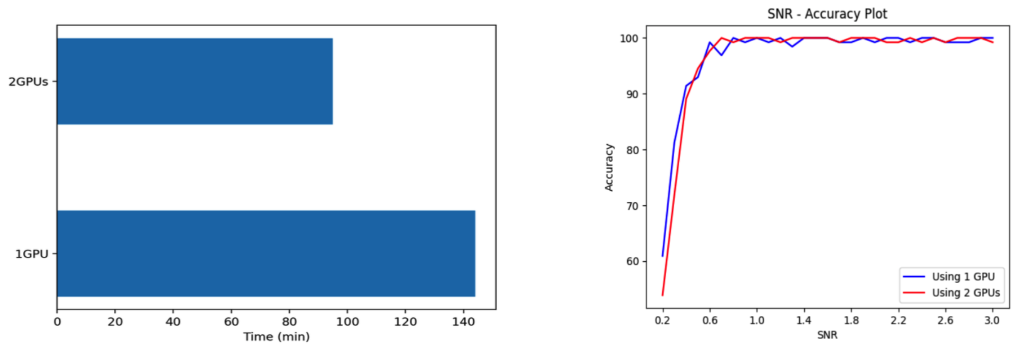


Figure 9. Result of multi-GPU approach. Left: comparison of time between 1GPU and 2GPUs training. Right: the test accuracy of 1GPU and 2GPUs training on our classifier.

4. Conclusion

In this project, our implementation of GW classifier has successfully classified the Gaussian noise and the Gaussian noise with GW signal. However, with similar deep neural network structure of the GW classifier, the GW predictor accuracy of solving the regression problem is not as good as we expect. In this project, our multi-GPU Training implementation also demonstrates how the multiple GPUs computing devices could accelerate the deep neural network learning process. Plus, this Multi-GPU Training approach could be the infrastructure for our future CNN with more hidden layers. From this project we experimented and learned how traditional signal processing methods, such as matching filtering, can be applied for GW detection and how the deep learning approaches can solve the same problem in different ways.

5. Acknowledgements

This project is one of the on-going projects at Deep Learning Team at Gravitational Wave Group at NCSA. We thank Prof. Zhizhen (Jane) Zhao and Daniel George for their guidance and help. We also express our gratitude to Dr. Eliu Huerta for granting us the access to Campus Cluster GPUs. We thank Zixin Huang and other members of the Deep Learning team for the insightful comments. This project is also submitted as the final project for CS598 PS course at University of Illinois at Urbana-Champaign in Fall 2017 semester. We thank Prof. Paris Smaragdis, TAs Cem Subakan, and Zesheng Wang for giving us a great opportunity to study machine learning and signal processing techniques.

References

[1] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, Adhikari, and *et al.*, “Observation of gravitational waves from a binary black hole merger,” *Phys. Rev. Lett.*, vol. 116, p. 061102, Feb 2016.

[2] B. Allen, J. K. Blackburn, P. R. Brady, J. D. E. Creighton, T. Creighton, S. Droz, A. D. Gillespie, S. A. Hughes, S. Kawamura, T. T. Lyons, J. E. Mason, B. J. Owen, F. J. Raab, M. W. Regehr, B. S. Sathyaprakash, R. L. Savage, S. Whitcomb, and A. G. Wiseman, “Observational limit on gravitational waves from binary neutron stars in the galaxy,” *Phys. Rev. Lett.*, vol. 83, pp. 1498–1501, Aug 1999.

[3] D. George and E. A. Huerta, “Deep Neural Networks to Enable Real-time Multimessenger Astrophysics,” 2016.

[4] D. George and E. A. Huerta, “Deep Learning for Real-time Gravitational Wave Detection and Parameter Estimation: Results with Advanced LIGO Data,” 2017.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. Software available from tensorflow.org.

[6] “Training a model using multiple gpu cards.” `https://www.tensorflow.org/tutorials/deep_cnn`. Accessed: 2017-11.

[7] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, “Accurate, large minibatch SGD: training imagenet in 1 hour,” *CoRR*, vol. abs/1706.02677, 2017.