













Scalable matched-filtering pipeline for gravitational-wave searches of compact binary mergers

Yun-Jing Huang ^{1,2,*} Chad Hanna,^{1,2,3,4} Becca Ewing,^{1,2} Patrick Godwin ⁵ Joshua Gonsalves ^{1,2,6} Ryan Magee ⁷ Cody Messick ⁸ Leo Tsukada ^{9,10} Zach Yarbrough ¹¹ Prathamesh Joshi ^{1,2} James Kennington ^{1,2} Wanting Niu ^{1,2} Jameson Rollins ⁵ and Urja Shah ¹²

¹*Department of Physics, The Pennsylvania State University, University Park, PA 16802, USA*

²*Institute for Gravitation and the Cosmos, The Pennsylvania State University, University Park, PA 16802, USA*

³*Department of Astronomy and Astrophysics, The Pennsylvania State University, University Park, PA 16802, USA*

⁴*Institute for Computational and Data Sciences, The Pennsylvania State University, University Park, PA 16802, USA*

⁵*LIGO Laboratory, California Institute of Technology, MS 100-36, Pasadena, California 91125, USA*

⁶*Department of Computer Science and Engineering,
The Pennsylvania State University, University Park, PA, 16802, USA*

⁷*LIGO Laboratory, California Institute of Technology, Pasadena, CA 91125, USA*

⁸*Leonard E. Parker Center for Gravitation, Cosmology, and Astrophysics,
University of Wisconsin-Milwaukee, Milwaukee, WI 53201, USA*

⁹*Department of Physics and Astronomy, University of Nevada,
Las Vegas, 4505 South Maryland Parkway, Las Vegas, NV 89154, USA*

¹⁰*Nevada Center for Astrophysics, University of Nevada, Las Vegas, NV 89154, USA*

¹¹*Department of Physics and Astronomy, Louisiana State University, Baton Rouge, LA 70803, USA*

¹²*School of Physics, Georgia Institute of Technology, Atlanta, GA 30332, USA*

(Dated: October 11, 2024)

As gravitational-wave observations expand in scope and detection rate, the data analysis infrastructure must be modernized to accommodate rising computational demands and ensure sustainability. We present a scalable gravitational wave search pipeline which modernizes the GstLAL pipeline by adapting the core filtering engine to the PyTorch framework, enabling flexible execution on both Central Processing Units (CPUs) and Graphics Processing Units (GPUs). Offline search results on the same 8.8 day stretch of public gravitational wave data indicate that the GstLAL and the PyTorch adaptation demonstrate comparable search performance, even with float16 precision. Lastly, computational benchmarking results show that the GPU float16 configuration of the PyTorch adaptation executed on an A100 GPU can achieve a speedup factor of up to 169 times compared to GstLAL’s performance on a single CPU core.

I. INTRODUCTION

Since the first detection of gravitational waves from a binary black hole merger, GW150914 [1], the LIGO [2], Virgo [3] and KAGRA [4][5] collaborations have completed three observing runs, and reported over 90 candidates in the third Gravitational-Wave Transient Catalog (GWTC-3) [6]. These consistent detections not only affirm gravitational-wave astronomy as an established field but also lead to groundbreaking discoveries, such as the first binary neutron star merger, GW170817 [7]. With upcoming ground-based and space-based detectors in multiple stages of development [8–10], we can expect a rapid increase in both data volume and detection rates, paving the way for new discoveries ahead.

As the scale of these observations expands, the computational demands on detection pipelines will increase drastically. GstLAL is a gravitational wave detection pipeline capable of processing data in low-latency [11–13]. It has played a key role in the detection of GW150914 [1], was the first pipeline to detect GW170817 in low-latency [7], and has been consistently making detections

throughout all the observing runs [6, 14–16]. GstLAL employs matched-filtering to search for compact binary coalescences (CBC) [17], a technique that is implemented in various forms in many search pipelines [18–20]. This technique is effective for identifying signals in noisy strain data, and requires filtering the data with a pre-generated bank of templates, typically containing $O(10^6)$ templates [21], which makes exploring high-dimensional parameter spaces computationally expensive.

Modern computational techniques, such as GPU acceleration and machine learning, have been explored for CBC searches [20, 22]. These approaches potentially provide scalability to handle increasing data size and allow for more extensive exploration of parameter spaces, leading to improved scientific insights. In this work, we adapt the core matched-filtering foundations of GstLAL to a modernized framework, PyTorch [23], and demonstrate that it can maintain detection efficiency while providing the robustness needed to meet computational demands.

We chose the PyTorch framework for several reasons. First, the PyTorch library supports GPU execution, offering higher computational power and memory bandwidth, which allows for scaling the filtering pipeline and expanding both the template bank and search parameter space. Second, PyTorch enables seamless switching

* yun-jing.huang@ligo.org

between CPU and GPU, supporting efficient use of both resources at computing centers while minimizing the need to maintain multiple code versions. Third, GPU support allows us to explore float16 precision, reducing computational costs and permitting further scaling of the template bank. Fourth, using Python, a high-level language, shortens development time, promoting faster progress and a broader developer and user community. Finally, PyTorch provides easy integration of machine learning techniques for future enhancements.

This paper is organized as follows: Section II outlines the workflow of the GstLAL pipeline and introduces its adaptation to PyTorch. In Section III, we present search results from public gravitational wave data using the PyTorch adaptation and compare them to the original GstLAL results. Section IV evaluates and benchmarks the computational performance of the PyTorch adaptation against GstLAL. Finally, in Section V, we discuss our conclusions and outline plans for future work.

II. PIPELINE DESCRIPTION

The GstLAL search pipeline can be divided into two operational modes: “online”, which processes data in low-latency, and “offline”, which processes archival gravitational wave data. In this paper, we will perform analyses using the “offline” mode of the GstLAL workflow. For a description of the “online” mode of the GstLAL pipeline for O4, see [12]. For a description of both online and offline search for O1 and O2, see [11].

A. GstLAL inspiral workflow

GstLAL uses time-domain matched-filtering to search for gravitational wave candidates in detector strain data [11, 17]. The process begins by correlating the data with a bank of templates, identifying potential candidates, and then assigning significance to each. The workflow can be broken down into several stages.

In the setup stage, the template bank for the matched-filtering process is generated. For the O4 run, the GstLAL template bank contains approximately $\sim 2 \times 10^6$ CBC templates [21]. These templates are divided into groups of ~ 1000 , and within each group, the Low-Latency Online Inspirial Detection (LLOID) algorithm is applied [17]. The templates are split into different time slices, which then undergo singular value decomposition (SVD) [24]. Each group of decomposed templates is referred to as an “SVD bank,” which is used in the next filtering stage.

In the filtering stage, the strain data is processed by first estimating the Power Spectral Density (PSD). The running average of the PSD is used to whiten the strain data, after which large deviations in the whitened data are gated and removed, completing the data conditioning process [11]. The conditioned data is then filtered using

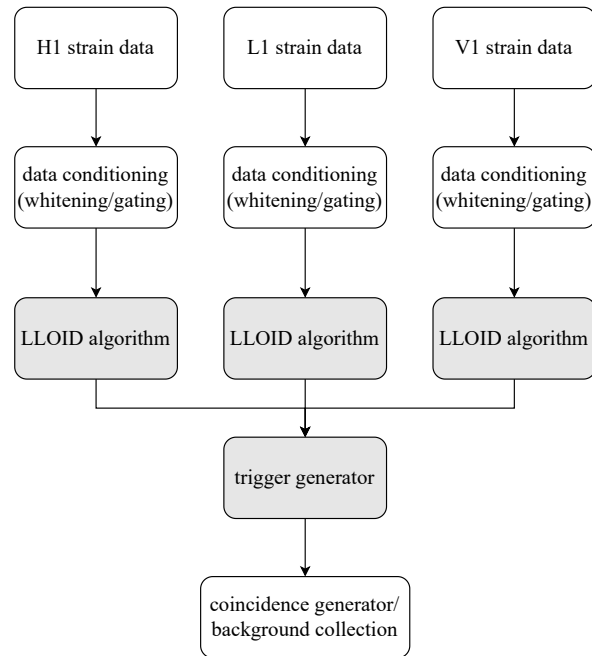


FIG. 1: Flowchart of the inspiral program in the GstLAL offline workflow. The shaded elements are the processes adapted to the PyTorch framework, and can be executed either on the CPU or GPU, offering flexibility and portability. The workflow begins with strain data from multiple detectors being read into the pipeline. The data is then conditioned through whitening and gating. Afterward, the conditioned data is processed using the LLOID algorithm, GstLAL’s implementation of matched-filtering [17]. The LLOID algorithm generates SNR time series, and SNR peaks with values ≥ 4 are identified as triggers in the trigger generator. These triggers are then sent to the coincidence generator to form coincident triggers, while non-coincident triggers during coincidence detector times are gathered as background data.

the SVD bank from the setup stage, producing signal-to-noise ratio (SNR) time series for each template. This is achieved using the LLOID algorithm, which down-samples the conditioned data, cross-correlates it with orthogonal templates, performs matrix multiplication of the orthogonal SNR time series with SVD coefficients, and finally up-samples and sums the physical SNR segments [17]. Once the LLOID algorithm produces the SNR time series, the inspiral program identifies peaks in the SNR time series with $\text{SNR} \geq 4$, which are defined as “triggers.” The phase and the signal consistency metric ξ^2 are then calculated for these triggers. These triggers are passed to the coincidence generator, which identifies coincidences between triggers detected with the same template and within the light travel time between detectors. Non-coincident triggers that occur within the coincident detector time are collected as background data.

In the injection stage, simulated CBC signals are generated based on a set of injection parameters, and these

waveforms are added to the strain data at specific time intervals. The combined data is then filtered using the same algorithm applied in the filtering stage.

Finally, in the ranking stage, significance is assigned to the triggers identified during filtering. Triggers are initially ranked using a ranking statistic, which is then used to determine their false alarm rate (FAR). GstLAL uses the likelihood ratio as the ranking statistics, calculated against the background data collected during the filtering stage [13, 25]. Triggers from the injection stage are also ranked against background data from the filtering stage. The fraction of injections that pass a given FAR threshold is then used to evaluate the performance of the pipeline.

B. PyTorch adaptation

The filtering stage of the GstLAL workflow, which filters whitened data with the template bank to produce SNR time series, is the primary computational bottleneck. In GstLAL, the LLOID algorithm is implemented as a set of Gstreamer [26] elements in C, and runs exclusively on the CPU [27]. In this work, we re-implement the LLOID algorithm using PyTorch.

The LLOID algorithm performs several linear algebra operations including cross-correlation, matrix multiplication, resampling, and addition. In the PyTorch adaptation, we replace these operations with PyTorch library functions and integrate them into the workflow. Additionally, we introduced an optimization that enables filtering across multiple SVD banks in parallel. Since SVD time slices with the same sample rate share the same input and output dimensions during filtering, these templates are grouped and stacked together during initialization. We then leverage PyTorch’s multi-batch and multi-channel operations to execute the filtering computations simultaneously across multiple SVD banks.

In addition to re-implementing the LLOID algorithm, the trigger generation stage was also adapted for PyTorch. This change was necessary because the LLOID algorithm generates SNR time series for each template in the bank, and transferring such a large dataset from the GPU to the CPU would create a substantial memory bandwidth bottleneck. By adapting the trigger generation to PyTorch, we can reduce the data transfer by creating a more refined dataset that is sent from the GPU to the CPU. The inspiral program in the GstLAL offline workflow and the PyTorch adaptation are shown in Figure 1.

III. O3 DATA SEARCH RESULTS

In this section, we aim to demonstrate the validity of the PyTorch adaptation by comparing its offline gravitational wave search results with those of GstLAL. For the PyTorch adaptation, we show search results under

three different configurations: (1) CPU with float32 precision, (2) GPU with float32 precision, and (3) GPU with float16 precision. First, we outline the dataset and search parameter space used for the analyses, followed by a comparison of the known gravitational wave events recovered by each search configuration. Finally, we evaluate the difference in injection recovery.

A. Dataset

We analyze an 8.8 day stretch of public HLV strain data obtained from the Gravitational Wave Open Science Center (GWOSC) [28]. The data cover the time range of May 12, 2019 19:36:42 UTC to May 21, 2019 14:45:08 UTC, which is during the third observing run (O3).

B. Search parameter space

The template bank used in the analyses targets the BBH region and covers a parameter space with component masses between 3 and 200 M_\odot , M_{chirp} between 6 and 200 M_\odot , mass ratio between one and ten, and spin z -components between -0.35 and 0.35. The frequency range for the matched-filter integration is from 15 to 1024 Hz. In total, there are 9,693 templates, which are grouped into 10 SVD banks.

C. Gravitational wave events

There are six gravitational wave events previously reported in GWTC-3 within the time span of the dataset [6]. Table I lists search results for the six gravitational wave events in the O3 dataset used for this study. Using the small BBH template bank, GstLAL and all three different configurations of the PyTorch adaptation recovered the six known gravitational wave events consistently.

The best match templates were the same among all four analyses for each of the six events. The PyTorch adaptation recovers slightly lower network SNRs than those of GstLAL for the events GW190513.205428, GW190514.065416, GW190521, with a relative difference of 0.2%, 0.2%, and 0.01%, respectively. This discrepancy can be attributed to the additional feature of sub-sample interpolation in the GstLAL pipeline during the SNR peak finding stage, which is not implemented in the PyTorch adaptation and is left for future development. The network SNRs for the other three events are consistent across all four analyses. The FAR values recovered by the analyses are also consistent. The results for GPU float16 show that performing matched-filtering in half precision can produce consistent results with those of float32 precision, and can be effectively used for detection in a gravitational wave search pipeline.

Figure 2 shows the inverse false-alarm rate (IFAR) plots for the four different analyses for the duration of the

search dataset. The dashed lines represent the expected count distribution from background noise. The observed distribution agree with the expected noise count at low IFAR and clearly branch out at six number of events, indicating the presence of the six known gravitational wave events in the dataset. All the plots in Figure 2 for all four of the analyses show consistent IFAR distributions. In particular, Figure 2d demonstrates that using float16 precision in a gravitational wave search pipeline is feasible.

D. Injection recovery

In addition to the strain data, we process an identical stretch of strain data with injection frames added to evaluate the relative performance difference of the four pipeline configurations. The injection set was generated with source frame component masses between 10 to 100 M_{\odot} , spin z -components between -0.3 to 0.3, and maximum redshift of 3. The injections are distributed uniformly in comoving volume. Injection frames were generated from the injection parameters and inserted into the strain data every 48 seconds, resulting in a total of 15,615 injections through out the duration of the dataset.

We next focus on comparing the relative number of found injections across the four analyses. An injection is considered “found” if it has a FAR value below a FAR threshold of one per thirty days. Table II lists the number of injections found for each of the analyses, categorized by the observing instrument times. The results demonstrate that the injection recovery of the PyTorch adaptation aligns closely with that of GstLAL. Both the CPU float32 version and GPU float32 version have a total number of found injections with a relative difference of less than 1% compared to GstLAL. The GPU float16 configuration, with a total number of detected events 1.13% lower than GstLAL, shows slightly reduced performance but remains comparable.

IV. COMPUTATIONAL PERFORMANCE

In this section, we seek to evaluate the computational performance for the different configurations of the PyTorch adaptation, using the GstLAL analyses as a baseline. The part of the pipeline that is benchmarked is the inspiral job, which begins with the ingestion of strain data, and ends with the generation of triggers. We perform a three detector search, with the strain data being three stretches of gaussian data colored with the PSDs of each of the HLV instruments. The SVD bank used has 1,004 templates, with template duration of 73 seconds. There is one SVD bank for each instrument, since the SVD templates need to be whitened by the PSDs of the instruments. When we process multiple banks in the following tests, we provide the same SVD bank multiple times.

The CPU benchmarking tests are conducted on a computing node equipped with AMD EPYC 7313 Processors, with 32 physical CPU cores in total. We launched 64 inspiral jobs on the computing node in parallel. Each job processes one hour of continuous HLV gaussian data, and one SVD bank for each instrument. We assess computational performance by using “templates in real-time” as a metric, which is defined as

$$\frac{\text{number of templates} \times \text{data duration (s)}}{\text{number of cores} \times \text{wall time (s)}} \quad (1)$$

and serves as a measure of how many templates the pipeline can process in real-time on a single CPU core.

The GPU benchmarking tests are performed on two kinds of GPUs, NVIDIA A2 16GB and NVIDIA A100-SXM4-80GB. As explained in Section IIB, the PyTorch adaptation will stack multiple SVD banks during the filtering stage and process the multiple SVD banks in parallel. Therefore, for benchmarking the GPU configurations, we run one inspiral job and vary the number of SVD banks the pipeline processes to evaluate the performance as a function of the number of SVD banks. We use the same SVD bank used for the CPU benchmarking tests. For the GPU “templates in real-time” metric, we consider the GPU as one processing unit, and set “number of cores” as one in Equation 1.

Figure 3 presents the performance of GstLAL and the different configurations of the PyTorch adaptation. The CPU version of PyTorch is slower than that of GstLAL, and its optimization is intended for future work. The performance of the GPU configurations improves as the number of SVD banks processed within one job increases. The performance grows linearly for small number of SVD banks, and slowly plateaus at larger number of SVD banks, when the GPU’s resources are saturated. Figure 4 shows the speedup factor of the PyTorch adaptation compared to GstLAL on a single CPU core. The GPU analyses clearly outperform GstLAL’s single CPU core performance, with the highest speedup factor being 169.14x.

V. CONCLUSION

In this paper, we present a scalable matched-filter based gravitational wave detection pipeline, where the core filtering engine and trigger generator of the GstLAL pipeline have been replaced with PyTorch modules. The flexibility to switch between computation devices allows us to offload computational bottlenecks to GPUs for enhanced performance. Additionally, we introduced a feature in the filtering algorithm that leverages PyTorch’s multi-batch and multi-channel options to filter multiple SVD banks in parallel.

To validate the detection performance of the PyTorch adaptation, we filtered GstLAL and three PyTorch configurations: CPU float32, GPU float32, and GPU float16,

TABLE I: Search results of the gravitational wave events previously reported in GWTC-3 within the dataset for GstLAL and three configurations of the PyTorch adaptation. The instruments listed in the “found inst.” column are those which identified the event with a trigger of $\text{SNR} \geq 4.0$. The SNRs in the table are network SNRs.

Name	Found Inst.	GstLAL		PyTorch adaptation					
		SNR	FAR (yrs^{-1})	CPU float32		GPU float32		GPU float16	
				SNR	FAR (yrs^{-1})	SNR	FAR (yrs^{-1})	SNR	FAR (yrs^{-1})
GW190513_205428	H1L1V1	12.30	4.92×10^{-6}	12.27	1.37×10^{-5}	12.27	4.49×10^{-6}	12.27	1.29×10^{-5}
GW190514_065416	H1L1	8.404	3.83	8.403	3.32	8.403	2.62	8.402	3.24
GW190517_055101	H1L1	10.01	0.0011	10.01	0.00095	10.01	0.0010	10.01	0.0016
GW190519_153544	H1L1	13.26	7.41×10^{-7}	13.26	6.80×10^{-7}	13.26	7.40×10^{-7}	13.26	8.18×10^{-7}
GW190521	H1L1	14.57	0.0010	14.54	0.0010	14.54	0.0011	14.54	0.0026
GW190521_074359	H1L1	23.55	3.13×10^{-27}	23.55	3.86×10^{-27}	23.55	3.46×10^{-27}	23.55	6.29×10^{-27}

TABLE II: Number of found injections for each of the four analyses, categorized by different combinations of observing instruments. The “Sum” row adds up all the injection numbers from all the different combinations.

On Inst.	GstLAL	PyTorch adaptation		
		CPU float32	GPU float32	GPU float16
H1L1V1	1996	2017	1990	1982
H1L1	989	989	978	966
H1V1	99	99	99	98
L1V1	330	333	332	327
H1	11	11	11	10
L1	77	78	78	79
V1	19	19	19	19
Sum	3521	3546	3507	3481

using an 8.8-day stretch of O3 public data. All four configurations successfully recovered the six known gravitational wave events with consistent network SNR and significance. Results from the injection campaign also indicate that injection recoveries were comparable across all analyses.

We benchmarked the PyTorch adaptation and compared its computational performance with GstLAL. Our results show that using the GPU float16 configuration on an A100 GPU, the PyTorch adaptation achieved up to 169 times the speed of GstLAL running on a single CPU core.

This study further demonstrates that float16 precision can yield comparable results to float32 precision, making it effective for detecting known gravitational wave events while offering better computational performance. This improvement in efficiency enables us to potentially expand the search parameter space.

In the future, we plan to modernize the remaining portions of the pipeline still using the Gstreamer framework. With the increased performance from GPU configurations, there is potential to further expand the search

parameter space. Additionally, leveraging the PyTorch framework will allow us to integrate machine learning techniques, explore alternative detection methods, and enhance the pipeline’s overall sensitivity.

ACKNOWLEDGMENTS

This material is based upon work supported by NSF’s LIGO Laboratory which is a major facility fully funded by the National Science Foundation. This research has made use of data, software and/or web tools obtained from the Gravitational Wave Open Science Center (<https://www.gw-openscience.org/>), a service of LIGO Laboratory, the LIGO Scientific Collaboration and the Virgo Collaboration. We especially made heavy use of the LVK Algorithm Library [29, 30]. LIGO Laboratory and Advanced LIGO are funded by the United States National Science Foundation (NSF) as well as the Science and Technology Facilities Council (STFC) of the United Kingdom, the Max-Planck-Society (MPS), and the State of Niedersachsen/Germany for support of the construction of Advanced LIGO and construction and operation of the GEO600 detector. Additional support for Advanced LIGO was provided by the Australian Research Council. Virgo is funded, through the European Gravitational Observatory (EGO), by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale di Fisica Nucleare (INFN) and the Dutch Nikhef, with contributions by institutions from Belgium, Germany, Greece, Hungary, Ireland, Japan, Monaco, Poland, Portugal, Spain.

The authors are grateful for computational resources provided by the the Pennsylvania State University’s Institute for Computational and Data Sciences gravitational-wave cluster, and the LIGO Lab cluster at the LIGO Laboratory, and supported by the National Science Foundation awards OAC-2103662, PHY-2308881, PHY-2011865, OAC-2201445, OAC-2018299, PHY-0757058, PHY-0823459, and PHY-2207728. CH Acknowledges generous support from the Eberly College of Science, the Department of Physics, the Institute for

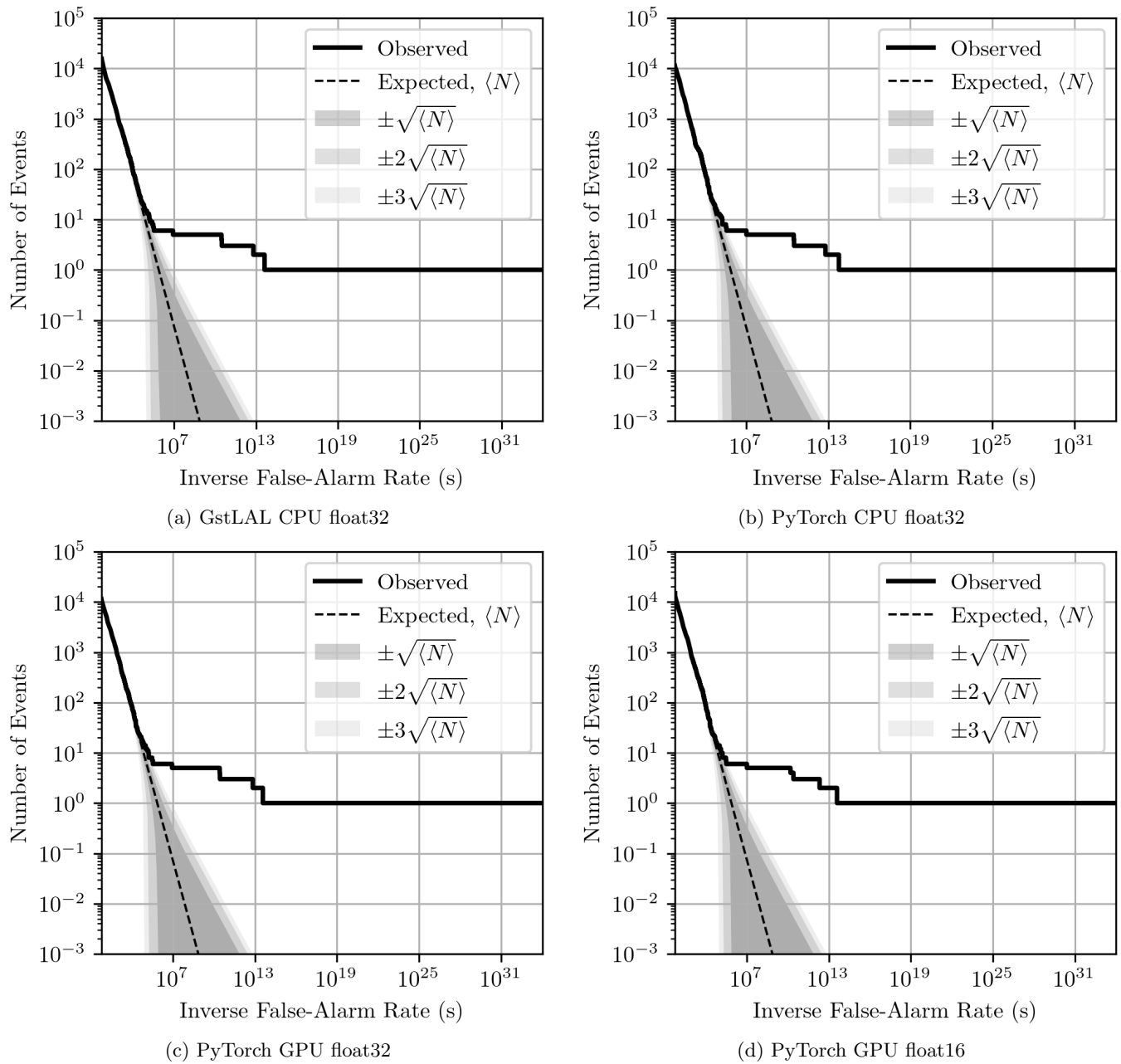


FIG. 2: Number of events vs. inverse false-alarm rate for the four analyses. The dashed lines are the expected number of events coming from background noise. The black lines are the number of events actually observed.

Gravitation and the Cosmos, and the Institute for Computational and Data Sciences.

-
- [1] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Observation of gravitational waves from a binary black hole merger, *Phys. Rev. Lett.* **116**, 061102 (2016).
- [2] J. Aasi *et al.* (LIGO Scientific Collaboration), Advanced ligo, *Classical and Quantum Gravity* **32**, 074001 (2015).
- [3] F. Acernese *et al.*, Advanced virgo: a second-generation interferometric gravitational wave detector, *Classical and Quantum Gravity* **32**, 024001 (2014).
- [4] T. Akutsu *et al.*, Overview of KAGRA: Detector design and construction history, *Progress of Theoretical and Experimental Physics* **2021**, 05A101 (2020).
- [5] KAGRA joined midway through the third observing run.
- [6] R. Abbott *et al.* (LIGO Scientific Collaboration, Virgo

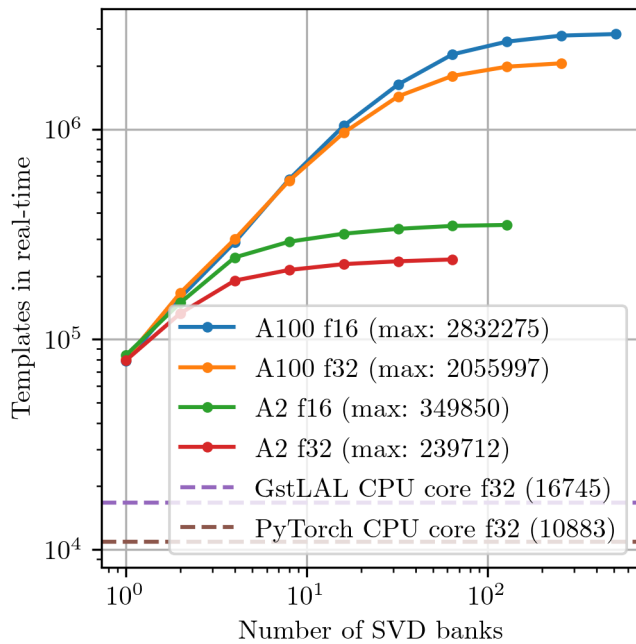


FIG. 3: Computational performance of the four analyses. For the PyTorch GPU configuration, two different GPUs were used for the benchmarking: NVIDIA A2 16GB (A2) and NVIDIA A100-SXM4-80GB (A100). The purple and brown dashed lines are the templates per CPU core benchmarks for GstLAL and PyTorch CPU configuration, respectively. Float16 and float32 precisions are represented as “f16” and “f32”, respectively. The numbers in the parenthesis are the maximum templates in real-time values for the GPU benchmarks, and the templates per CPU core in real-time values for the CPU benchmarks.

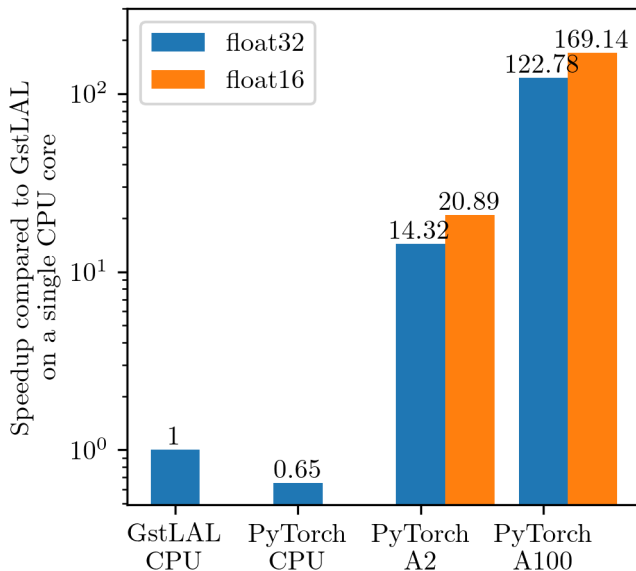


FIG. 4: Speedup factor of different device and data type configurations of the PyTorch adaptation compared to GstLAL on a single CPU core. The blue bars represent the configurations with float32 precision, whereas the orange bars represent those with float16 precision.

- Collaboration, and KAGRA Collaboration), Gwtc-3: Compact binary coalescences observed by ligo and virgo during the second part of the third observing run, *Phys. Rev. X* **13**, 041039 (2023).
- [7] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Gw170817: Observation of gravitational waves from a binary neutron star inspiral, *Phys. Rev. Lett.* **119**, 161101 (2017).
- [8] S. Dwyer, D. Sigg, S. W. Ballmer, L. Barsotti, N. Mavalvala, and M. Evans, Gravitational wave detector with cosmological reach, *Phys. Rev. D* **91**, 082001 (2015).
- [9] M. Punturo *et al.*, The einstein telescope: a third-generation gravitational wave observatory, *Classical and Quantum Gravity* **27**, 194002 (2010).
- [10] P. Amaro-Seoane *et al.*, Laser interferometer space antenna (2017), arXiv:1702.00786 [astro-ph.IM].
- [11] C. Messick *et al.*, Analysis framework for the prompt discovery of compact binary mergers in gravitational-wave data, *Phys. Rev. D* **95**, 042001 (2017).
- [12] B. Ewing *et al.*, Performance of the low-latency gstlal inspiral search towards ligo, virgo, and kagra’s fourth observing run, *Phys. Rev. D* **109**, 042008 (2024).
- [13] L. Tsukada *et al.*, Improved ranking statistics of the gstlal inspiral search for compact binary coalescences, *Phys. Rev. D* **108**, 043004 (2023).
- [14] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Gwtc-1: A gravitational-wave transient catalog of compact binary mergers observed by ligo and virgo during the first and second observing runs, *Phys. Rev. X* **9**, 031040 (2019).
- [15] R. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Gwtc-2: Compact binary coalescences observed by ligo and virgo during the first half of the third observing run, *Phys. Rev. X* **11**, 021053 (2021).
- [16] R. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Gwtc-2.1: Deep extended catalog of compact binary coalescences observed by ligo and virgo during the first half of the third observing run (2022), arXiv:2108.01045 [gr-qc].
- [17] K. Cannon, R. Cariou, A. Chapman, M. Crispin-Ortuzar, N. Fotopoulos, M. Frei, C. Hanna, E. Kara, D. Keppel, L. Liao, S. Privitera, A. Searle, L. Singer, and A. Weinstein, Toward early-warning detection of gravitational waves from compact binary coalescence, *The Astrophysical Journal* **748**, 136 (2012).
- [18] F. Aubin, F. Brighenti, R. Chierici, D. Estevez, G. Greco, G. M. Guidi, V. Juste, F. Marion, B. Mours, E. Nitoglia, O. Sauter, and V. Sordini, The mbta pipeline for detecting compact binary coalescences in the third ligo–virgo observing run, *Classical and Quantum Gravity* **38**, 095004 (2021).
- [19] T. D. Canton, A. H. Nitz, B. Gadre, G. S. C. Davies, V. Villa-Ortega, T. Dent, I. Harry, and L. Xiao, Real-time search for compact binary mergers in advanced ligo and virgo’s third observing run using pycbc live, *The Astrophysical Journal* **923**, 254 (2021).
- [20] Q. Chu, M. Kovalam, L. Wen, T. Slaven-Blair, J. Bosveld, Y. Chen, P. Clearwater, A. Codoreanu, Z. Du, X. Guo, X. Guo, K. Kim, T. G. F. Li, V. Oloworaran, F. Panther, J. Powell, A. S. Sengupta, K. Wette, and X. Zhu, Spiir online coherent pipeline to search for gravitational waves from compact binary coalescences, *Phys. Rev. D* **105**, 024023 (2022).
- [21] S. Sakon *et al.*, Template bank for compact binary merg-

- ers in the fourth observing run of advanced ligo, advanced virgo, and kagra, *Phys. Rev. D* **109**, 044066 (2024).
- [22] E. Marx, W. Benoit, A. Gunny, R. Omer, D. Chatterjee, R. C. Venterea, L. Wills, M. Saleem, E. Moreno, R. Raikman, E. Govorkova, D. Rankin, M. W. Coughlin, P. Harris, and E. Katsavounidis, A machine-learning pipeline for real-time detection of gravitational waves from compact binary coalescences (2024), arXiv:2403.18661 [gr-qc].
- [23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, Pytorch: An imperative style, high-performance deep learning library (2019), arXiv:1912.01703 [cs.LG].
- [24] K. Cannon, A. Chapman, C. Hanna, D. Keppel, A. C. Searle, and A. J. Weinstein, Singular value decomposition applied to compact binary coalescence gravitational-wave signals, *Phys. Rev. D* **82**, 044025 (2010).
- [25] K. Cannon, C. Hanna, and J. Peoples, Likelihood-ratio ranking statistic for compact binary coalescence candidates with rate estimation (2015), arXiv:1504.04632 [astro-ph.IM].
- [26] Gstreamer, <https://gstreamer.freedesktop.org/documentation/index.html>.
- [27] K. Cannon, S. Caudill, C. Chan, B. Cousins, J. D. Creighton, B. Ewing, H. Fong, P. Godwin, C. Hanna, S. Hooper, R. Huxford, R. Magee, D. Meacher, C. Messick, S. Morisaki, D. Mukherjee, H. Ohta, A. Pace, S. Privitera, I. de Ruiter, S. Sachdev, L. Singer, D. Singh, R. Tapia, L. Tsukada, D. Tsuna, T. Tsutsui, K. Ueno, A. Viets, L. Wade, and M. Wade, Gstlal: A software framework for gravitational wave discovery, *SoftwareX* **14**, 100680 (2021).
- [28] R. Abbott *et al.* ((The LIGO Scientific Collaboration, the Virgo Collaboration, and the KAGRA Collaboration)), Open data from the third observing run of ligo, virgo, kagra, and geo, *The Astrophysical Journal Supplement Series* **267**, 29 (2023).
- [29] LSC Algorithm Library software packages LAL, LALWRAPPER, and LALAPPS.
- [30] LIGO Scientific Collaboration, LIGO Algorithm Library - LALSuite, free software (GPL) (2018).