DEEP LEARNING METHODS FOR ENABLING REAL-TIME GRAVITATIONAL WAVE AND MULTIMESSENGER ASTROPHYSICS

BY

DANIEL GEORGE

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Astronomy in the Graduate College of the University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Professor Gabrielle D. Allen, Chair and Advisor
Research Assistant Professor Eliu A. Huerta
Professor H. Edward Seidel
Assistant Professor Zhizhen Zhao
Professor Brian D. Fields
Professor Bernard F. Schutz
Abstract

A new era of gravitational wave (GW) astronomy has begun with the recent detections by LIGO. However, we need real-time observations of GW signals and their electromagnetic (EM) and astro-particle counterparts to unlock its full potential for scientific discoveries. Extracting and classifying the wide range of modeled and unmodeled GWs, whose amplitudes are often much weaker than the background noise, and rapidly inferring accurate parameters of their source is crucial in enabling this scenario of real-time multimessenger astrophysics. Identifying and automatically clustering anomalous non-Gaussian transient noises (glitches) that frequently contaminate the data and separating them from true GW signals is yet another difficult challenge.

Currently, the most sensitive data analysis pipelines are limited by the extreme computational costs of template-matching methods and thus are unable to scale to all types of GW sources and their full parameter space. Accurate numerical models of GW signals covering the entire range of parameters including eccentric and spin-precessing compact binaries, which are essential to infer the astrophysical parameters of an event, are not available. Searches for unmodeled and anomalous signals do not have sufficient sensitivity compared to the targeted searches. Furthermore, existing search pipelines are not optimal for dealing with the non-stationary, non-Gaussian noise in the detectors. This indicates that many critical events will go unnoticed. The primary objective of this thesis is to resolve these issues via deep learning, a state-of-the-art machine learning method based on artificial neural networks.

In this thesis we develop robust GW analysis algorithms for analyzing real LIGO/Virgo
data based on deep learning with neural networks, that overcomes many limitations of existing techniques, allowing real-time detection and parameter estimation modeled GW sources and unmodeled GW bursts as well as classification and unsupervised clustering of anomalies and glitches in the detectors. This pipeline is designed to be highly scalable, therefore it can be trained with template banks of any size to cover the entire parameter-space of eccentric and spin-precessing black hole binaries as well as other sources and also optimized based on the real-time characteristics of the complex noise in the GW detectors.

This deep learning framework may also be extended for low-latency analysis of the raw big data collected across multiple observational instruments to further facilitate real-time multimessenger astrophysics, which promises groundbreaking scientific insights about the origin, evolution, and destiny of the universe. In addition, this work introduces a new paradigm to accelerate scientific discovery by using data derived from high-performance physics simulations on supercomputers to train artificial intelligence algorithms that exploit emerging hardware architectures.
To everyone who taught, supported, and inspired me along the way
# Table of Contents

**List of Abbreviations** ................................................................. 1

**Chapter 1  Introduction** .............................................................. 2

**Chapter 2  Deep Learning for Gravitational Wave Detection and Parameter Estimation** .......................................................... 12
  2.1 Introduction .............................................................................. 13
  2.2 Neural Networks and Deep Learning ........................................ 17
    2.2.1 Artificial neural networks ................................................. 19
    2.2.2 Convolutional neural networks ......................................... 21
    2.2.3 Time-series analysis with convolutional neural networks .... 22
  2.3 Method ...................................................................................... 25
    2.3.1 Assumptions ......................................................................... 26
    2.3.2 Obtaining data ....................................................................... 27
    2.3.3 Designing neural networks ................................................. 29
    2.3.4 Training strategy .................................................................. 32
  2.4 Results ...................................................................................... 34
    2.4.1 Detection ............................................................................... 34
    2.4.2 Parameter estimation .......................................................... 37
    2.4.3 New classes of gravitational wave sources ....................... 39
    2.4.4 Speed and computational cost ............................................ 43
  2.5 Discussion .................................................................................. 45
  2.6 Conclusion .................................................................................. 49
  2.7 Appendix .................................................................................... 51
    2.7.1 Preparing training and testing data ..................................... 51
    2.7.2 Designing and training neural networks ............................ 54
    2.7.3 Comparisons with other methods ....................................... 56
    2.7.4 Measuring accuracy and errors ......................................... 59

**Chapter 3  Application of Deep Learning with Real Advanced LIGO Data** .... 65
  3.1 Introduction ............................................................................... 66
  3.2 Methods ...................................................................................... 68
  3.3 Results ....................................................................................... 73
  3.4 Conclusion .................................................................................. 78
  3.5 Acknowledgements ..................................................................... 80
List of Abbreviations

NR  Numerical Relativity
LIGO Laser Interferometer Gravitational Wave Observatory
DNN Deep Neural Network
ANN Artificial Neural Network
CNN Convolutional Neural Network
RNN Recurrent Neural Network
GW Gravitational Wave
eLISA Evolved Laser Interferometer Space Antenna
WFIRST Wide Field Infrared Survey Telescope
JWST James Webb Space Telescope
ADAM Adaptive Moment Estimation
BBH Binary Black Hole
BNS Binary Neutron Star
NSBH Neutron Star Black Hole
IMBH Intermediate-Mass Black Hole
GRB Gamma-Ray Burst
EM Electromagnetic
HPC High-Performance Computing
AI Artificial Intelligence
Chapter 1

Introduction

Gravitational wave (GW) science has recently revolutionized observational astronomy and theoretical astrophysics. During the first observing run of the advanced Laser Interferometric Gravitational wave Observatory (aLIGO) detectors, we peered into the most powerful events in the cosmos driven by extreme gravitational interactions (Abbott et al., 2016; Abbott et al., 2016a). These direct detections of GWs from the mergers of binary black hole (BBH) systems provided the first glimpse of the nature of gravity in the strong-field regime (Abbott et al., 2016; Abbott et al., 2016a, 2017b,a). It offered the earliest observational evidence for the formation and merger of BBHs within a Hubble time, confirmed the existence of massive stellar-mass BHs, and provided the first direct measurement of the angular momentum of BHs (Abbott et al., 2016b; Belczynski et al., 2010; Antonini et al., 2016; Abbott et al., 2016a; Rodriguez et al., 2016; Belczynski et al., 2016; Marchant et al., 2016; de Mink & Mandel, 2016; Abbott et al., 2016a).

The European advanced Virgo (aVirgo) detector (Acernese et al., 2015) joined the aLIGO detectors during the second observing run. The recent detection of the binary black hole (BBH) merger (GW170814) with this three-detector network enabled new phenomenological tests of general relativity regarding the nature of GW polarizations, while significantly improving the sky localization of this GW transient (Abbott et al., 2017a). This enhanced capability to localize GW transients provided critical input for the first detection of GWs from the merger of two neutron stars (NSs) and in conjunction with follow-up observations across the electromagnetic (EM) spectrum (Abbott et al., 2017b). This multimessenger event has finally confirmed that NS mergers are the central engines of short gamma-ray
bursts (GRBs) and produce half of all elements heavier than iron (Abbott et al., 2017b; Eichler et al., 1989; Paczynski, 1986; Narayan et al., 1992; Kochanek & Piran, 1993; LIGO Scientific Collaboration et al., 2017; Piran et al., 2013; Lee et al., 2010; Lee & Ramirez-Ruiz, 2007; Ott, 2009; Phinney, 2009). We expect that future detections of neutron star-black hole mergers may confirm whether these systems also produce short GRBs, and whether rapidly rotating hypernovae are the progenitors of long duration GRBs, collapsars, etc. (Ott, 2009; Phinney, 2009).

After this series of breakthroughs, it is evident that the detection of GWs will become a common occurrence as the aLIGO and Virgo detectors gradually attain design sensitivity in the following years. Furthermore, a worldwide network of kilometer-scale GW detectors in the US, Europe, and Asia will considerably increase the science reach of GW astrophysics (The LIGO Scientific Collaboration et al., 2015; Acernese et al., 2015; Hirose et al., 2014; Unnikrishnan, 2013).

High-Performance Computing (HPC) plays a central role in GW astrophysics as these studies relied on accurate massively-parallel numerical relativity (NR) simulations of Einstein’s field equations for sources of GWs such as black holes, neutron stars, and supernovae. These simulations served a dual purpose: (i) they conclusively showed that Einstein’s general relativity can describe with outstanding accuracy the true nature of gravity in the most extreme astrophysical environments and (ii) having thus established its status, comparison with signal templates generated or calibrated with NR is extensively used to extract GW signals from the highly noisy data streams and infer astrophysical properties of the detected GW sources (The LIGO Scientific Collaboration et al., 2016).

Through GWs alone we can observe exotic events that do no emit light and probe the structure of the Universe at large scales, the behavior of gravity in the strong-field regime for stringent tests of general relativity, get better insights into stellar evolution processes, the mechanisms of formation and evolution of stellar-mass and supermassive black holes, cosmic strings, the stochastic GW background, and possibly the nature of Dark Matter and
Dark Energy (Sathyaprakash & Schutz, 2009; Abbott et al., 2016b). However, the emphasis of multimessenger astrophysics will be on extraordinary events, such as the mergers of binary neutron stars (BNSs), NSBH, intermediate-mass black holes (IMBHs) with masses between $100M_\odot - 500M_\odot$, IMBHs and stellar mass BHs or NSs, as well as eccentric binary coalescences, core-collapse supernovae, and other exotic unexpected events. Joint discovery campaigns with powerful optical detectors such as the Dark Energy Survey (DES) (The Dark Energy Survey Collaboration, 2005), the Large Synoptic Survey Telescope (LSST) (LSST Science Collaboration et al., 2009), Euclid (Amendola et al., 2013), and WFIRST (Spergel et al., 2013) will enable the coincident detection of GW events which are expected to generate electromagnetic (EM) counterparts. Through coincident observations with other messengers such as photons, neutrinos, and cosmic rays that we can achieve groundbreaking scientific insights (Abbott et al., 2016b). When we simultaneously observe events across the entire GW, electromagnetic (EM), and astro-particle spectrums, we will gain an unprecedented understanding of the interactions between the fundamental forces, origin of gamma-ray bursts, equation of state of extremely dense matter, physics at the core of supernovae, and effects at the boundary of quantum field theory and strong-field gravity, thus bringing us closer to a grand unified theory of our universe (Christensen et al., 2011; Smith et al., 2013).

Therefore, the future of astronomy is multimessenger astrophysics in which we expect to hear an event first through our GW detectors and send out immediate alerts to astronomers, who then immediately turn their telescopes around to see it within seconds, and also feel it via neutrino, cosmic-ray, and other astro-particle detectors. These real-time multimessenger observations will help us understand the origin, evolution, and destiny of the universe. Bringing multimessenger astrophysics into fruition requires the combination of multi-spectrum observations with very low latency. Detecting a GW candidate in real-time and establishing whether its parameters correspond to likely progenitors of EM counterparts (Eichler et al., 1989; Paczynski, 1986; Narayan et al., 1992; Kochanek & Piran, 1993; Piran et al., 2013; Lee et al., 2010; Lee & Ramirez-Ruiz, 2007) is essential to enable rapid broadband EM and
astro-particle searches (Röver et al., 2009; Littenberg et al., 2016). Obtaining the complete information immediately after such events from multiple messengers can provide unique insights into their dynamics which are inaccessible otherwise (Ott, 2009; Phinney, 2009). For example, if GWs from the inspiral of binary neutron stars can be identified and their sky localization estimated by a real-time search algorithm tens of seconds before merger, telescopes may be slewed to see the full kilonova explosion for the first-time since previously we have only observed the afterglow several hours after the merger (Abbott et al., 2017b).

There are several challenges in realizing this scenario of real-time multimessenger astrophysics:

• **Existing methods to extract weak GW signals are inadequate** as the most sensitive searches rely on matched-filtering, repeatedly comparing data streams against over 300,000 templates (Usman et al., 2016a). Even with thousands of CPUs, these flagship searches for modeled sources are restricted to a small subset i.e. quasi-circular and spin-aligned compact binaries. Recent studies indicate that moderately eccentric BBH populations may be missed by existing quasi-circular GW searches (Huerta et al., 2017; Tiwari et al., 2016; Huerta et al., 2014, 2017b). The computational complexity of matched-filtering explodes exponentially with the number of templates. Therefore, targeting the full parameter-space of astrophysically motivated sources is computationally infeasible even with exascale resources. Accurately reconstructing the parameters of an event takes several days (Veitch et al., 2015a), which is often too late for precise EM follow-up campaigns. Furthermore, these searches are only optimal when the background noise is Gaussian, whereas the noise in the detectors is non-Gaussian and non-stationary.

• **Anomalous non-Gaussian transients, known as glitches, frequently contaminate GW detector data.** Since their high occurrence rate in LIGO and Virgo data can obscure or even mimic true GW signals, successfully identifying and excising these anomalies is of utmost importance for several reasons (Powell et al., 2017; Zevin et al., 2015b).
a) This will prevent false GW detections due to coincident glitches across multiple detectors that closely mimic signals. b) Rapidly identifying and excising glitches will enhance detector sensitivity, and improve the significance of GW signals that are contaminated by glitches (Abbott et al., 2017b). c) The LIGO and Virgo detectors have thousands of instrumental and environmental channels to monitor changes that occur due to environmental or hardware issues. By carefully tracking down these glitches, we aim to identify their source and eliminate them promptly to ensure that the data stream is usable for GW data analysis. More types of glitches are expected to be generated as the detectors undergo modifications to improve their sensitivity. Classifying existing known classes of glitches and automatically clustering new classes of glitches as they arise in the future, is a difficult task requiring human intervention through citizen science campaigns or the development of new “intelligent” algorithms, because of their complex morphologies (Zevin et al., 2017).

There is a lack of GW templates from numerical relativity simulations covering the full range of expected sources, especially eccentric and spin-precessing compact binaries. GW astronomy relies on template banks calibrated via accurate massively-parallel numerical relativity simulations of Einstein’s field equations to carry out detection and parameter estimation of GW signals. Without accurate GW template banks of all types of sources, many weak signals would be missed by existing GW pipelines, as we showed in (Huerta et al., 2016). Thousands of additional numerical relativity simulations are required to even sparsely sample the large space of parameters and each one is expensive to compute, requiring weeks of run-time on supercomputers. Therefore search algorithms that can automatically generalize beyond the few available templates to interpolate and, more importantly, extrapolate to new regions in the parameter space and new types of GW sources (such as supernovae) are needed.

Finding an EM counterpart by scanning a large region of the sky, due to
the poor localization of GW detectors (Abbott et al., 2016b), and rapidly classifying them into potential sources of a GW event (from short, highly noisy, exposures) is yet another difficult challenge, requiring rapid generation of more accurate sky maps as well as new image processing and search techniques capable of intelligently analyzing the huge influx of raw big data collected by various types of telescopes in low-latency and potentially on-site.

Artificial Intelligence (AI), based on deep learning with artificial neural networks (loosely modeled after biological neurons) (Goodfellow et al., 2016; Lecun et al., 2015), offers an ideal framework to tackle these challenges. Deep learning is rapidly becoming a ubiquitous technology that is revolutionizing every industry today and is now being widely applied for academic research. Deep learning thrives on big data and has recently achieved dramatic successes, in every field of AI ranging from image/speech recognition and synthesis, to complex games (e.g. Poker, Go), to self-driving vehicles, to health care, to natural language understanding and translation (Schmidhuber, 2015; Silver et al., 2016; Najafabadi et al., 2015). Deep learning is known for its scalability, i.e., the ability to take advantage of petascale or larger datasets. While HPC is now an essential component of modern science, the full potential of applying deep learning and AI for research in the fundamental sciences, such as Physics and Astronomy, remains largely untapped. The main focus of this PhD thesis is on developing AI methods based on deep learning algorithms that can exploit emerging hardware architectures such as GPUs, for improved signal processing and analysis of data from GW detectors in order to enable real-time GW and multimessenger astrophysics.

These methods and their application to LIGO data are presented in the following chapters, each of which are based on first-author peer-reviewed publications by the author and they are presented in a self-contained manner. In Chapter 2, we start with an overview of deep learning and then introduce Deep Filtering, a new deep learning-based method we developed for directly processing raw time-series data using a combination of Deep Convolutional Neural Networks, which enables real-time detection, classification, and parameter
estimation of GW signals hidden in extremely noisy time-series data. This approach is particularly appealing for LIGO data analysis as it diverts all the intensive computation to a \textit{one-time} training process, therefore allowing us to scale to template banks of any size and still perform analysis rapidly with minimal computational resources. In Chapter 3, we demonstrate how this technique can be applied to real data collected from the first observing run of advanced LIGO. Here, we show for the first time that machine learning can detect true GW signals.

To summarize, we demonstrate the following advantages of using deep learning over existing search algorithms in these two chapters:

- **Speed**: The analysis can be carried out within milliseconds using deep learning with minimal computational resources, i.e., a single CPU/GPU, which will help generate alerts to enable rapid follow-up observations of electromagnetic counterparts, which can lead to new physical insights.

- **Covering more parameter-space**: Only a small subset of the full parameter space of signals can be searched for using matched filtering (template matching), since the computational cost explodes exponentially with the number of parameters. On the other hand, Deep learning is highly scalable and requires only a one-time training process, so the full high-dimensional parameter space can be covered with existing computational resources.

- **Automatic generalization to new sources**: The article shows that signals from new classes of sources beyond the training data, such as spin-precessing or eccentric compact binaries, can be automatically detected with this method with the same sensitivity. This is because, unlike template-matching techniques, deep learning can interpolate to points within the training data and generalize beyond it to some extent. Detection of such GW sources, especially those with eccentricity, is important as they can provide us more insight into astrophysical processes in globular clusters and galactic nuclei [Huerta].
• **Resilience to real non-Gaussian noise**: The results show that this deep learning method can distinguish signals from transient non-Gaussian noises (glitches) and can accurately detect and estimate parameters of GW signals even when they are contaminated by glitches, unlike matched filtering. For instance, the occurrence of a glitch in coincidence with the recent detection of the neutron star merger led to a missed trigger in the existing pipelines, which delayed the analysis by several hours and was detected only after required manual inspection. We show the deep learning technique can automatically detect events obscured by very loud glitches and even estimate their parameters with high accuracy.

• **Acceleration of existing pipelines, interpretability**: Once the deep learning method detects a signal and predicts its parameters, this can be quickly cross-validated using matched filtering with a few templates around these predicted parameters. The estimates of the astrophysical properties of GW candidates returned by this method can be used as a starting point to speed up more informative Bayesian parameter estimation methods while providing rough estimates to start instant EM follow-up campaigns. Therefore, deep learning can be combined with matched-filtering in a hierarchical search to accelerate traditional GW detection pipelines and extend their range of parameters by narrowing down their search space, so that the interpretability of the results is not lost.

In Chapter 4, we present *Deep Transfer Learning* as a new method for classifying glitches and for automatically clustering new anomalous signals and noises in the LIGO/Virgo data. By leveraging pre-trained state-of-the-art convolutional neural networks designed for real-world object recognition, and fine-tuning them throughout all layers on a small dataset of LIGO spectrograms, we develop an improved glitch classification algorithm. Then we show that by truncating these neural networks, thus transferring their knowledge about the
known classes of LIGO data and real-world images, we can use them as feature-extractors for clustering (grouping together) entirely new classes of anomalies in an unsupervised manner.

The main results of this chapter are highlighted below:

- We show that this technique achieves state-of-the-art results for glitch classification, attaining above 98.8% overall accuracy and perfect precision-recall on 8 out of 22 classes, while significantly reducing the training time to a few minutes. This reduces the error rate by more than a factor of 2 compared to the previous best model (Zevin et al., 2016; Bahaadini et al., 2017). This is a dramatic increase since improvements in classification accuracy become more difficult to attain as they tend towards 100%. Our method is able to achieve more or less the same performance as citizens scientists.

- The application of this new method will be useful to narrow down the source of specific glitches by correlating their occurrences with measurements from various instruments. By finding and eliminating these glitches we can increase the duty cycle of GW detectors during upcoming discovery campaigns, a critical factor to increase the number of new detections.

- Our transfer learning method enables the accurate classification of classes of glitches given very limited labeled examples, unlike previous approaches (Zevin et al., 2016; Bahaadini et al., 2017). This is of utmost importance for rapidly learning to classify new classes of glitches in upcoming discovery campaigns.

- Reduces the training time by several orders of magnitude. Once trained, our method can analyze data streams faster than real-time. Furthermore, our method eliminates the need for designing new neural network architectures and manually optimizing their hyper-parameters for this specific glitch classification task.

---

1Hyperparameters refer to several quantities that have to be manually chosen to determine the architecture of the neural networks (e.g., overall design, number of layers, sizes of the convolutions, depth, padding, types of layers and activations, etc.)
- The neural networks we use were originally designed for classifying over 1000 different classes of real-world objects in images. Therefore, our algorithm can be easily extended to classify hundreds of new classes of glitches in the future, especially since this transfer learning approach requires only a few labeled examples of a new class.

- We show how entirely new classes of anomalies can be automatically grouped together by truncating our CNNs, trained on known classes, and then using them as feature extractors for unsupervised or semi-supervised clustering algorithms in combination with dimension reduction algorithms. This implies that we may also use this technique for GW searches by looking for the occurrence of the same cluster of anomalies in multiple detectors at the same time.

- Our results show that features for real-world object detection can be transferred directly for classifying and clustering time-series data by converting them to spectrogram images, even though the two datasets are very dissimilar. This technique may be useful for analyzing time-series datasets in general for anomaly detection, classification, and clustering in other disciplines.

In the final chapter, we briefly describe some of our ongoing work on the application of deep learning for denoising GW data, for detecting of GWs from other types of sources such as supernovae, for generating improved sky localization maps, etc. We conclude this thesis by suggesting how the great flexibility and computational efficiency offered by our deep learning methods could promote them as a standard tool for multimessenger astrophysics and beyond.
Deep Learning for Gravitational Wave Detection and Parameter Estimation

Gravitational wave astronomy has set in motion a scientific revolution. To further enhance the science reach of this emergent field of research, there is a pressing need to increase the depth and speed of the algorithms used to enable these groundbreaking discoveries. We introduce Deep Filtering—a new scalable machine learning method for end-to-end time-series signal processing. Deep Filtering is based on deep learning with two deep convolutional neural networks, which are designed for classification and regression, to detect gravitational wave signals in highly noisy time-series data streams and also estimate the parameters of their sources in real-time. Acknowledging that some of the most sensitive algorithms for the detection of gravitational waves are based on implementations of matched-filtering, and that a matched-filter is the optimal linear filter in Gaussian noise, the application of Deep Filtering using whitened signals in Gaussian noise is investigated in this foundational article. The results indicate that Deep Filtering outperforms conventional machine learning techniques, achieves similar performance compared to matched-filtering, while being several orders of magnitude faster, allowing real-time signal processing with minimal resources. Furthermore, we demonstrate that Deep Filtering can detect and characterize waveform signals emitted from new classes of eccentric or spin-precessing binary black holes, even when trained with datasets of only quasi-circular binary black hole waveforms. The results presented in this article, and the recent use of deep neural networks for the identification of optical transients in telescope data, suggests that deep learning can facilitate real-time searches of gravitational wave sources and their electromagnetic and astro-particle coun-

terparts. In the subsequent article, the framework introduced herein is directly applied to identify and characterize gravitational wave events in real LIGO data.

2.1 Introduction

Gravitational wave (GW) astrophysics is a well established field of research. To date, the advanced Laser Interferometer Gravitational wave Observatory (aLIGO) detectors \cite{The LIGO Scientific Collaboration & The Virgo Collaboration 2016; The LIGO Scientific Collaboration et al. 2015} have detected five GW events from binary black hole (BBH) mergers that are consistent with Einstein’s general relativity predictions \cite{Abbott et al. 2016; Abbott et al. 2016a; Abbott et al. 2017b; Abbott et al.}

By the end of aLIGO’s second discovery campaign (O2), the European advanced Virgo (aVirgo) detector \cite{Acernese et al. 2015} joined aLIGO, establishing the first, three-detector search for GW sources in the advanced detector era. This international network was critical for the detection of the fifth BBH merger with improved sky localization, and also provided the means to carry out new phenomenological tests of gravity \cite{Abbott et al. 2017a}.

The international aLIGO-aVirgo detector network was used for the first detection of GWs from two colliding neutron stars (NSs), GW170817 \cite{Abbott et al. 2017b; Abbott et al. 2016a; Singer et al. 2014}, which was followed up with broadband electromagnetic observations after several hours \cite{LIGO Scientific Collaboration et al. 2017}. These multime- senger observations led to the first direct confirmation that NS mergers are the progenitors of gamma rays bursts, GRB170817A \cite{Abbott et al. 2017; Eichler et al. 1989; Paczynski 1986; Narayan et al. 1992; Kochanek & Piran 1993; Ott 2009; Phinney 2009}, and the cosmic factories where about half of all elements heavier than iron are produced \cite{LIGO Scientific Collaboration et al. 2017}. These major scientific breakthroughs, worthy of the 2017 Nobel Prize in Physics, have initiated a new era in contemporary astrophysics.

Ongoing improvements in the sensitivity of aLIGO and aVirgo, will enable future multi-
messenger observations with astronomical facilities Dark Energy Survey Collaboration et al. (2016); Abdo et al. (2013); Tyson (2002); Amendola et al. (2013); Gehrels et al. (2015); ANTARES Collaboration et al. (2016), increasing the number and types of GW sources, and providing new and detailed information about the astrophysical origin, and cosmic evolution of compact objects.

Multimessenger astrophysics is an interdisciplinary program that brings together experimental and theoretical physics, cosmology, fundamental physics, high performance computing (HPC) and high throughout computing (HTC). For instance, at the interface of HPC and theoretical physics, numerical relativity (NR) simulations of Einstein’s field equations are extensively used to validate the astrophysical nature of GW sources Abbott et al. (2016b); Healy et al. (2017). Furthermore, NR simulations of NS mergers, neutron star-black hole (NSBH) mergers, core collapse supernovae and other massive, relativistic systems provide key physical insights into the physics of GW sources that are expected to generate electromagnetic (EM) and astro-particle counterparts Ott (2009); Mösta et al. (2014); Haas et al. (2016); Abdikamalov et al. (2014); Kidder et al. (2017); Nissanke et al. (2013).

On the other hand, large scale GW data analysis has traditionally relied on HTC resources. Flagship GW searches have been very successful at exploiting these resources to identify and characterize GW sources Usman et al. (2016b); Cannon et al. (2012); Abbott et al. (2016a); Cornish & Littenberg (2015). Within the next few years GW discovery campaigns will bring together an international network of GW interferometers, that will gather data for extended periods of time. As the sensitivity of this detector network reaches design sensitivity, the detection rate will continue to increase in successive detection campaigns. Furthermore, existing low latency (online) matched-filtering based algorithms currently target only a 4-dimensional (4D) parameter space, which describes spin-aligned compact binary sources.

Accelerating the offline Bayesian parameter estimation algorithms, which typically last

---

1 Astrophysically motivated sources describe a 9-dimensional intrinsic parameter space: two component masses, eccentricity, and two 3D vectors describing the spin of each binary component.
from several hours to a few days, is no trivial task since they have to sample a 15D parameter space Smith et al. (2016); Veitch et al. (2015b); Harry et al. (2016); Littenberg et al. (2016).

In light of these challenges, there are ongoing efforts to reduce the size of template banks used for matched-filtering based GW searches Indik et al. (2017). Based on these considerations, and realizing that to maximize the science one can extract from GW observations, it is essential to rapidly cover a deeper parameter space of astrophysically motivated sources, the GW community has been exploiting state-of-the-art HPC facilities to increase the pool of computational resources to carry out for large scale GW data analysis Huerta et al. (2017a).

To further contribute to fully realize the multimessenger astrophysics program, this article introduces a new machine (deep) learning algorithm, Deep Filtering, which is based on deep neural networks (DNNs) Lecun et al. (2015) to directly process highly noisy time-series data for both classification and regression in real-time. In particular, this algorithm consists of two deep convolutional neural networks LeCun & Bengio (1998) that take time-series inputs, and are capable of detecting and estimating parameters of GW signals whose peak power is weaker than that of the background noise.

The main objective in developing Deep Filtering is to complement and enhance the existing, low latency GW detection algorithms, such as PyCBC Usman et al. (2016b) and gstLAL Messick et al. (2017), to enable deeper and faster GW searches. Deep Filtering may be applied to identify and rapidly constrain the astrophysical parameters of GW transients. This real-time analysis would then be followed up by existing offline Bayesian parameter estimation pipelines Smith et al. (2016); Veitch et al. (2015b). A targeted search of this nature can significantly reduce the size of multi-dimensional template banks, enabling the use of established matched-filtering searches at a fraction of their computational cost to quantify the significance of new GW detections. This approach would combine the best of two approaches: the scalable nature of DNNs with the sophistication of LIGO-Virgo detection pipelines.

In this foundational article, we describe the key features of Deep Filtering and carry out
a systematic study of DNNs trained using a dataset of inspiral-merger-ringdown (IMR) BBH waveforms [Taracchini et al. (2014); Bohé et al. (2017)] to cover the BBH parameter-space where ground-based GW detectors are expected to have the highest detection rate [Belczynski et al. (2016)]. This analysis is carried out using GW signals whitened with aLIGO’s design sensitivity injected into Gaussian noise. This simplified scenario is studied in this first article to illustrate the key ideas and new deep learning methods in a transparent manner, and also to compare these results to a matched-filter, the optimal linear filter in Gaussian noise, which is at the core of some of the most sensitive GW detection pipelines [Usman et al. (2016b); Cannon et al. (2012)]. In the subsequent article, the methods presented here are successfully applied for the detection and characterization of GW signals in real LIGO data [George & Huerta (2017)].

The results in this article suggest that DNNs may be ideal tools for enhancing GW analysis. In particular, DNNs are able to interpolate between waveform templates, in a similar manner to Gaussian Process Regression (GPR) and to generalize to some new classes of signals beyond the templates used for training. An important advantage of Deep Filtering is its scalability, i.e., all the intensive computation is diverted to the one-time training stage, after which the datasets can be discarded, i.e., the size of the template banks presents no limitation when using deep learning. With existing computational resources on supercomputers, such as Blue Waters, it will be feasible to train DNNs that target a 9D parameter space within a few weeks. Furthermore, once trained these DNNs can be evaluated in real-time with a single CPU, and more intensive searches over longer time periods covering a broader range of signals can be carried out with a dedicated GPU.

The analysis presented here, contextualized with recent work to understand and characterize aLIGO non-Gaussian noise transients [Zevin et al. (2016); George et al. (2018)], and new deep learning applications for transient identification in large sky surveys [Sedaghat] is a statistical tool that can serve as a probabilistic interpolation algorithm providing information about the training set of NR simulations needed to accurately describe a given parameter-space and generates interpolated waveforms that match NR counterparts above any given level of accuracy.
Mahabal (2017) suggests that it is feasible to create an efficient deep learning pipeline to perform all tasks—identifying the presence or absence of GW signals, classifying noise transients, reconstructing the astrophysical properties of detected GW sources, and identification of EM counterparts of GW events, thus paving a natural path to realizing real-time multimessenger astrophysics with a unified framework.

The application of deep learning in GW astrophysics, astronomy, and astro-particle physics has the potential to accelerate scientific research and unlock new opportunities by enhancing the way we use existing High Performance Computing (HPC) resources while allowing us to exploit emerging hardware architectures such as deep-learning-optimized Graphics Processing Units (GPUs) Chetlur et al. (2014) and Field-Programmable Gate Arrays (FP-GAs) Zhang et al. (2015). Working in tandem with computer scientists and industries to develop Artificial Intelligence (AI) tools that extend our prototype, and further exploring applications of deep learning for multimessenger astrophysics and fundamental sciences, may provide the means to effectively consolidate different windows of observation into the Universe.

This article is organized as follows: Section 2.2 provides a comprehensive overview of artificial neural networks and deep learning, particularly focusing on convolutional neural networks in the context of time-series signal processing. Section 3.2 describes the assumptions, datasets, and procedure to construct the DNN-based GW analysis pipeline. The results are reported in Section 3.3. In Section 2.5 the immediate implications for GW astrophysics missions are discussed. We summarize the findings and outline its broader applications in Section 3.4.

## 2.2 Neural Networks and Deep Learning

This section presents a brief overview of the main concepts of deep learning, including machine learning, artificial neural networks, and convolutional neural networks in the context
of time-series signal processing.

The vast majority of algorithms are designed with a specific task in mind. They require extensive modifications before they can be re-used for any other task. The term machine learning refers to a special class of algorithms that can learn from examples to solve new problems without being explicitly re-programmed. This enables cross-domain applications of the same algorithm by training it with different data [Goodfellow et al. (2016)]. More importantly, some of these algorithms are able to tackle problems which humans can solve intuitively but find difficult to explain using well-defined rules, hence they are often called “artificial intelligence” [Goodfellow et al. (2016)].

The two main categories of machine learning are supervised and unsupervised learning. In supervised learning, the algorithm learns from some data that is correctly labeled, while unsupervised learning algorithms have to make sense of unstructured and unlabeled data [Schmidhuber (2015)]. This work focuses on an application of supervised learning, where labeled data obtained from physics simulations is used to train an algorithm to detect signals embedded in noise and also estimate multiple parameters of the source.

Although traditional machine learning algorithms have been successful in several applications, they are limited in their ability to deal directly with raw data. Often the data has to be simplified manually into a representation suitable for each problem. Determining the right representation is extremely difficult and time-consuming, often requiring decades of effort even for domain experts, which severely limits the applicability of these algorithms [Goodfellow et al. (2016)]. Representation learning is a field of machine learning which aims to resolve this issue by creating algorithms that can learn by themselves to find useful representations of the raw data and extract relevant features from it automatically for each problem [Bengio et al. (2013)].

Deep Learning is one of the most rapidly growing subfields of machine learning, which resolves this difficulty of feature engineering with algorithms that can find useful representations of the raw data by extracting multiple levels of relevant features automatically for each
problem. This is achieved by combining a computational architecture containing long inter-
connected layers of “artificial neurons” with powerful learning (optimization) algorithms [Le-
cun et al. (2015); Goodfellow et al. (2016)]. These deep artificial neural networks (DNNs)
are able to capture complex non-linear relationships in the data by composing hierarchical
internal representations, all of which are learned automatically during the training stage.
The deepest layers are able to learn highly abstract concepts, based on the simpler outputs
of the previous layers, to solve problems that previously required human-level intelligence
thus achieving state-of-the-art performance for many tasks [Schmidhuber (2015)].

Deep learning powers many of the technologies routinely used by us including search en-
gines (Google, Bing), voice recognition on smartphones, personal assistants (Siri, Cortana,
Google assistant), smartphone keyboards, real-time face detection on cameras, face recogni-
tion (Facebook), language translation (Google Translate), text-to-speech synthesis [van den
Oord et al. (2016)], recommendations on Amazon, and automatic captioning on YouTube, to
name a few [Najafabadi et al. (2015)]. Most notably, deep learning was used in combination
with reinforcement learning [Sutton & Barto (1998)] to build a program called AlphaGo [Silver
et al. (2016)] which defeated one of the world’s best players, in 2016, at the highly complex
game of Go. Yet another recent success was at lip reading, where an algorithm has surpassed
the best humans by a large margin of accuracy [Son Chung et al. (2016)]. Deep learning is
also the key ingredient in self-driving vehicles that are being deployed across the world.

2.2.1 Artificial neural networks

Artificial neural networks (ANN), the building blocks of DNNs, are biologically-inspired
computational models that have the capability to learn from observational data [Nielsen
(2016)]. The fundamental units of neural networks are artificial neurons (loosely modeled
after real neurons [Graupe (2013)]), which are based on perceptrons introduced by Rosenblatt
in 1957 [Rosenblatt (1958)]. A perceptron takes a vector of inputs ($\vec{x}$) and computes a weighted
output with an offset known as bias. This can be modeled by the equation $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$,
where the weights ($\vec{w}$) and bias ($b$) are learned through training.

Minsky and Papert showed that a single perceptron has many limitations Minsky & Papert (1969). However, it was later found that these limitations can be overcome by using multiple layers of inter-connected perceptrons to create ANNs Schmidhuber (2015). The universality theorem Hornik et al. (1989) proves that ANNs with just three layers (one hidden layer) can model any function up to any desired level of accuracy.

Multilayer perceptrons are also known as feed-forward neural networks because information is propagated forward from the input layer to the output layer without internal cycles (i.e. no feedback loops) Goodfellow et al. (2016). While potentially more powerful cyclic architectures can be constructed, such as Recurrent Neural Networks Goodfellow et al. (2016) (RNNs), they are often more computationally expensive to train. Therefore, only feed-forward neural networks are considered in this article.

An ANN usually has an input layer, one or more hidden layers, and an output layer (shown in Figure 2.1). A non-linear “activation” function is applied to the output of each of the hidden layers. Without this non-linearity, using multiple layers would become redundant, as the network will only be able to express linear combinations of the input. The most commonly used non-linear activation functions are the logistic sigmoid, hyperbolic tan, and the rectified linear unit (also called ReLU or ramp). It has been empirically observed that the ramp produces the best results for most applications Jarrett et al. (2009). This function is mathematically expressed as $\text{max}(0, x)$.

The key ingredient that makes ANNs useful is the learning algorithm. Almost all neural networks used today are trained with variants of the back-propagation algorithm in conjunction with the gradient descent methods Schmidhuber (2015). The idea is to propagate errors backward from the output layer to the input layer after each evaluation of a neural network, in order to adjust the weights of each neuron so that the overall error is reduced in a supervised learning problem LeCun et al. (1998c). The weights of an ANN are usually initialized randomly to small values and then back-propagation is performed over multiple
2.2.2 Convolutional neural networks

A convolutional neural network (CNN), whose structure is inspired by studies of the visual cortex in mammals, is a type of feed-forward neural network. First developed by Fukushima for his Neocognitron, they were successfully combined with back-propagation by LeCun in the 1980s, for developing a highly accurate algorithm for recognizing handwritten digits. The exceptional performance of Alex Krizhevsky’s entry based on CNNs, which won the ImageNet competition by a huge margin in 2012, has sparked the current interest in these networks especially in the field of computer vision. CNNs have been found to approach or even surpass human-level accuracy at a variety of image and video...
processing tasks such as hand-writing recognition, identifying objects in photos, tracking movements in videos etc. Krizhevsky et al. (2012); Lecun et al. (2015).

The introduction of a “convolution layer”, containing a set of neurons that share their weights, is the critical component of these networks. Multiple convolution layers are commonly found in DNNs, with each having a separate set of shared weights that are learned during training. The name comes from the fact that an output equivalent to a convolution, or sometimes cross-correlation Goodfellow et al. (2016), operation is computed with a kernel of fixed size. A convolutional layer can also be viewed as a layer of identical neurons that each “look” at small overlapping sections of the input, defined as the receptive field.

The main advantage of using these layers is the ability to reduce computational costs by having shared weights and small kernels, thus allowing deeper networks and faster training and evaluation speeds. Because of the shared weights, CNNs are also able to automatically deal with spatially translated as well as (with a few modifications Lecun et al. (2015)) rotated and scaled signals. In practice, multiple modules each consisting of a sequence of convolution and pooling (sub-sampling) layers, followed by a non-linearity, are used. The pooling layers further reduce computational costs by constraining the size of the DNN, while also making the networks more resilient to noise and translations, thus enhancing their ability to handle new inputs Lecun et al. (2015). Dilated convolutions Yu & Koltun (2016) is a recent development which enables rapid aggregation of information over larger regions by having gaps within each of the receptive fields. In this study, we focus on CNNs as they are the most efficient DNNs on modern hardware, allowing fast training and evaluation (inference).

2.2.3 Time-series analysis with convolutional neural networks

Conventional methods for digital signal processing such as matched-filtering (cross-correlation or convolution against a set of templates) Owen & Sathyaprakash (1999) in time-domain or frequency-space are limited in their ability to scale to a large parameter-space of signal
templates, as discussed in Indik et al. (2017); Harry et al. (2016), while being too computationally intensive for real-time parameter estimation analyses Smith et al. (2016). Signal processing using machine learning in the context of GW astrophysics is an emerging field of research Graff et al. (2012); Mukund et al. (2017); Powell et al. (2017, 2015); Zevin et al. (2016); Bahaadini et al. (2017); George et al. (2018); Shen et al. (2017). These traditional machine learning techniques, including shallow ANNs, require “handcrafted” features extracted from the data as inputs rather than the raw noisy data itself. DNNs, on the other hand, are capable of extracting these features automatically.

Deep learning has been previously applied for the classification of glitches with spectrogram images as inputs to CNNs Zevin et al. (2016); Bahaadini et al. (2017); George et al. (2018) and unsupervised clustering of transients George et al. (2018), in the context of aLIGO. Using images as inputs is advantageous for two reasons: (i) there are well established architectures of 2D CNNs which have been shown to work (GoogLeNet Szegedy et al. (2015), VGG Simonyan & Zisserman (2014), ResNet He et al. (2015)); and (ii) pre-trained weights are available for them, which can speed up the training process via transfer learning while also providing higher accuracy even for small datasets George et al. (2018). However, experiments showed that this approach would not be optimal for detection or parameter estimation since many signals having low signal-to-noise ratio (SNR) are not visible in spectrograms, as shown in Fig. 2.2.

Theoretically, all the information about the signal is encoded within the time-series, whereas spectrograms are lossy non-invertible representations of the original data. Although 2D CNNs are commonly used, especially for image-related tasks, by directly feeding the time-series data as inputs to 1D CNNs, one can obtain higher sensitivities of detection (defined as the fraction of signals detected with respect to the total number of signals present in the inputs) at low SNR, lower error rates in parameter estimation, and faster analysis speeds.

3Note that the standard definition of optimal matched-filtering SNR is used in this article, as described in Owen & Sathyaprakash (1999). This SNR is on average proportional to $12.9 \pm 1.4$ times the ratio of the amplitude of the signal to the standard deviation of the noise for the test set.

4The error on the test set is defined as the mean of the magnitudes (absolute values) of the relative error.
Figure 2.2: **Sample of input data.** The red time-series is an example of the input to the deep neural network algorithm. It contains a binary black hole gravitational waveform signal (blue), which was whitened with aLIGO’s design sensitivity and superimposed in noisy data with SNR = 7.5 (peak power of this signal is 0.36 times the power of background noise). The component masses of the merging black holes are $57 M_\odot$ and $33 M_\odot$, respectively. The corresponding spectrogram on the right shows that the gravitational wave signal on the left is not visible, and thus cannot be detected by an algorithm trained for image recognition. Nevertheless, the deep neural network detects the presence of this signal directly from the (red) time-series input with over 99% sensitivity, and reconstructs the source’s parameters with a mean relative error of about 10%.

This automated feature learning allows the algorithm to develop more optimal strategies of signal processing than when given hand-extracted information such as spectrograms. There has been a few attempts at signal processing using CNNs with raw noisy time-series data in other domains which considered estimation of a single parameter [O’Shea et al. (2016); Zheng et al. (2014)].

This article demonstrates that DNNs can be used for both signal detection and multiple-parameter estimation directly from highly noisy time-series data, once trained with templates of the expected signals, and that deep CNNs outperform many traditional machine learning algorithms shown in Fig. 2.14 and reach accuracies comparable to matched-filtering methods. The results show that deep learning is more computationally efficient than matched-filtering for GW analysis. Instead of repeatedly performing overlap computations against all templates of known signals, the CNN builds a deep *non-linear* hierarchical structure of nested convolutions, with small kernels, that determines the parameters in a single evaluation estimating each parameter averaged over all inputs in the test set and over each parameter.
tion. Moreover, the DNNs act as an efficient compression mechanism by learning patterns and encoding all the relevant information in their weights, analogous to a reduced-order model [Pürrer (2016)], which is significantly smaller than the size of the training templates. Therefore, the DNNs automatically perform an internal optimization of the search algorithm and can also interpolate, or even extrapolate, to new signals not included in the template bank (unlike matched-filtering).

Note that matched-filtering performs the convolution of the input data against a set of templates, therefore, it is equivalent to a single convolution layer in a neural network, with very long kernels corresponding to each signal in the template bank. Therefore, Deep Filtering can be viewed as a more efficient extension of matched-filtering, which performs template matching against a small set of short duration templates, which are learned automatically, and aggregates this information in the deeper layers to effectively model the full range of long-duration signals.

### 2.3 Method

As a proof of concept in this first article, we focus on GWs from BBH mergers, which are expected to dominate the number of GW detections with ground-based GW detectors [Belczynski et al. (2016, 2015); Abbott et al. (2016c)]. Note that this method can be extended to GW signals produced by other types of events by adding more neurons in the final layer corresponding to the number of classes/parameters, changing the size of the input layer depending on the length of the templates, and training with template banks of these GW signals injected into simulated or real noise.

We have divided the problem into two separate parts, each assigned to a different DNN, so that they may be used independently. The first network, henceforth known as the “classifier”, will detect the presence of a signal in the input, and will provide a confidence level for the detection. The classes chosen for now are “True” or “False” depending on whether or not
a signal from a BBH merger is present in the input. The second network, referred to as the “predictor”, will estimate the parameters of the source of the signal (in this case, the component masses of the BBH). The predictor is triggered when the classifier identifies a signal with a high probability.

The system is partitioned in this manner so that, in the future, more classes of GW transients [Haas et al. (2016); Löffler et al. (2012); Mösta et al. (2014)], may be added to the classifier, and separate specialized predictors can be made for each type of signal. Moreover, categories for various types of anomalous sources of noise, like glitches and blips [Zevin et al. (2016); Cornish & Littenberg (2015)], can also be incorporated in the classifier [George et al. (2018)].

2.3.1 Assumptions

For this initial study, the signals are assumed to be optimally oriented with respect to the detectors, and that the individual spins and orbital eccentricities are zero. This reduces the parameter space to two dimensions, namely, the individual masses of the BBH systems, which is restricted to lie between $5M_\odot$ and $75M_\odot$. Furthermore, the inputs were constrained to have a duration of 1 second, and a sampling rate of 8192Hz throughout this analysis, which was an arbitrary choice made initially, which was found to perform well for the type of events that are considered here. Note that the classifier will be applied to the continuous data stream by using a sliding window of width 1 second. However, it is straightforward to use inputs of any duration by changing a hyperparameter corresponding to the input size of the CNNs, which will result in the computational cost scaling linearly with the length of the input.

Throughout this analysis, the signals were whitened using aLIGO’s Power Spectral Density (PSD) at the “Zero-detuned High Power” design sensitivity, shown in Figure 2.3, to approximate the sensitivity of LIGO at different frequencies. Consideration of transient sources of detector noise are deferred to the subsequent article. This is in line with previous
Figure 2.3: Sensitivity curve of aLIGO. Throughout this analysis, the Zero Detuned High Power sensitivity configuration for aLIGO was used to simulate the colored noise in the detectors by whitening the GW signals. The Amplitude Spectral Density (ASD) of the noise vs frequency for this configuration is shown in the figure.

studies, which have first showcased a machine learning algorithm for LIGO data analysis using simulated noise Powell et al. (2015); Veitch et al. (2015b); Torres-Forné et al. (2016), and then followed up by an independent study where the algorithm is tested using real aLIGO noise Powell et al. (2017). In this article, we follow a similar approach by describing the key concepts and methods for the construction of DNNs for GW data analysis in the context of Gaussian noise, and then show in the following article how this Deep Filtering algorithm can be directly applied to detect and characterize GW events in real LIGO data, which has non-Gaussian and non-stationary noise including glitches George & Huerta (2017).

2.3.2 Obtaining data

Supervised deep learning algorithms are more effective when trained with large datasets Goodfellow et al. (2016). Obtaining high quality training data has been a difficult and cumbersome task in most applications of DNNs, such as object recognition in images, speech and text processing, etc. Fortunately, this issue is not faced here since one can take advantage of scientific simulations to produce the necessary data for training.

Over the last decade, sophisticated techniques have been developed to perform accurate 3D NR simulations of merging BHs Mroué et al. (2013); Löffler et al. (2012). For the analysis
Figure 2.4: Distribution of data. The figure shows the distribution of component masses of BBHs for the training and testing datasets. The mass-ratios were confined between 1 and 10, which accounts for the missing points in the lower right corner. This mass-ratio range was chosen because the state-of-the-art EOB model used to create the datasets has only been validated for these mass-ratio values. Each point represents a quasi-circular, non-spinning GW signal of 1 second duration, sampled at 8192 Hz, which is whitened with aLIGO’s expected noise spectrum at design sensitivity. These waveforms were normalized and translated randomly in time. Thereafter, multiple batches of noise at each SNR were added to produce training and testing datasets.

at hand, Effective-One-Body (EOB) [Taracchini et al. (2014); Bohé et al. (2017)] waveforms that describe GWs emitted by quasi-circular, non-spinning BBHs are used. The final 1 second window of each template was extracted for this analysis.

Following the standard practice in machine learning, the data is split into separate sets for training and testing. For the training dataset, the BBHs component masses are in the range $5M_\odot$ to $75M_\odot$ in steps of $1M_\odot$. The testing dataset has intermediate component masses, i.e., masses separated from values in the training dataset by $0.5M_\odot$. By not having overlapping values in the training and testing sets, one can ensure that the network is not overfitting, i.e., memorizing only the inputs shown to it without learning to generalize to new inputs. The distribution of component masses, and a template from the training and testing sets, is shown in Fig. 2.4.
Subsequently, the location of the peak of each signal was shifted randomly within an interval of 0.2 seconds in both the training and testing sets to make the DNNs more robust with respect to time translations. Next, different realizations of Gaussian white noise were superimposed on top of the signals over multiple iterations, thus amplifying the size of the datasets. The power of the noise was adjusted according to the desired SNR for each training session. As usual, the inputs were standardized to have zero mean and unit variance to make the training process easier [LeCun et al., 1998b].

The final training sets at each SNR were produced from \(\sim 2500\) templates of GWs from BBH mergers by adding multiple batches of noise and shifting in time. It is also a standard practice to use a validation set to monitor the performance on unseen data during training in order to prevent overfitting. The validation and testing sets at each SNR were generated from a different set of \(\sim 2500\) templates by superimposing different noise realizations.

### 2.3.3 Designing neural networks

Similar DNN architectures were used for both the classifier and predictor, which demonstrates the versatility of this method. The only difference was the addition of a softmax layer to the classifier to obtain probability estimates as the outputs. The strategy was to first train the predictor on the datasets labeled with the BBH masses, and then transfer the weights of this pre-trained network to initialize the classifier and then train it on datasets in which half of the inputs contained an injected signal. This transfer learning process reduced the training time required for the classifier, while also slightly improving its accuracy at low SNR.

Overall, we designed and tested around 80 configurations of DNNs ranging from 1 to 4 convolutional layers and 1 to 3 fully connected layers (also called linear layers) similar to [LeCun et al., 1998a], but modified for time-series inputs. Among these, a design for the classifier with 3 convolutional layers followed by 2 fully connected layers yielded good results with fastest inference speed. We tried adding a few recent developments such as
Figure 2.5: **Architecture of deep neural network.** This is the deep dilated 1D CNN, modified to take time-series inputs, designed for prediction, which outputs two real-valued numbers for the two component masses of the BBH system. For classification, a *softmax* layer was added after the 14th layer to obtain the probabilities for two classes, i.e., “True” or “False”. The input is the time-series sampled at 8192Hz and the output is either the probability of each class or the value of each parameter. Note that the number of neurons in layer 14 can be increased to add more categories for classification or more parameters for prediction. The size of this CNN is about 2MB.

Batch normalization [Ioffe & Szegedy (2015a)] and dropout [Srivastava et al. (2014)] layers. However, they were not used in the final design as they did not provide improvements for the simple problem that is considered here. Note that the addition of noise to the signals during the training process serves as a form of regularization in itself. Many of the layers have parameters, commonly known as hyperparameters, which were tuned manually via a randomized trial-and-error procedure.

Depth is a hyperparameter which determines the number of filters in each convolutional layer. The choices for depth in the consecutive layers were 16, 32, and 64 respectively. Kernel sizes of 16, 8, and 8 were used for the convolutional layers and 4 for all the (max) pooling layers. Stride, which specifies the shift between the receptive fields of adjacent neurons, was chosen to be 1 for all the convolution layers and 4 for all the pooling layers. Dilation determines the overall size of each receptive field, which could be larger than the kernel size by having gaps in between. Here, it is a measure of the temporal extend of the convolutions. Using
Figure 2.6: **Architecture of deeper neural network.** This is the deeper version of the CNN, modified to take time-series inputs, designed for parameter estimation. The input is the time-series sampled at 8192Hz and the output is the predicted value of each parameter. This can be converted to a classifier by adding a *softmax* layer after layer 19 to obtain the probability for a detection. Note that the number of neurons in layer 19 can be increased to add more categories for classification or more parameters for prediction. The 2 neurons in the final layer outputs the 2 parameters corresponding to the individual masses of BBHs. The size of this CNN is approximately 23MB.
dilation of 4 in the final two convolution layers improved the performance. The final layout of the classifier DNN is shown in Fig. 2.5.

Deeper networks are expected to provide further improvements in accuracy although at the cost of slower evaluation speed. To show this, we also designed a deeper net, shown in Fig. 2.6 with 4 convolution layers and 3 fully connected layers that had improved sensitivity for detection and significantly better performance for parameter estimation. Although this design performed slightly better, it was a factor of 5 slower on a GPU for evaluation. This CNN had convolution layers having kernel sizes were 16, 16, 16, and 32 with dilations 1, 2, 2, and 2 respectively. The pooling layers all had kernel size 4 and stride 4.

A loss function (cost function) is required to compute the error after each iteration by measuring how close the outputs are with respect to the target values. A new loss function, i.e., the mean absolute relative error loss, was applied for training the predictor. For classification, the standard cross-entropy loss function (Goodfellow et al. 2016) was used.

### 2.3.4 Training strategy

Hyperparameter optimization was performed by trial and error to design architectures of the CNNs that achieved the best performance in terms of speed and accuracy. First, we used Gaussian white noise without whitening the signals i.e., a flat PSD, to determine the optimal architectures of the DNNs. This design was also found to be optimal for signals whitened with the Zero-Detuned PSD of aLIGO. This indicates that the same architecture will perform well on wide variety of PSDs. Once the best performing DNNs were chosen, they were trained for a total of approximately 10 hours. The DNNs were designed and trained using the neural network functionality in the Wolfram Language, Mathematica, based internally on the open-source MXNet framework (Chen et al. 2015), which utilizes the CUDA deep learning library (cuDNN) (Chetlur et al. 2014) for acceleration using GPUs. The ADAM (Kingma & Ba 2014) method as the learning algorithm. A snapshot of the training process is shown in Figure 2.7.
Figure 2.7: **Visualization of training.** This is a snapshot of one of the training sessions for parameter estimation. The mean squared error on the training set is plotted in orange and the blue curve measures the error on the validation set.

A new strategy was devised to reduce the training time of DNNs, while also ensuring an optimal performance, by starting off training the predictor on inputs having high SNR ($\geq 100$) and then gradually increasing the noise in each subsequent training session until a final SNR distribution randomly sampled in the range $5 \leq \text{SNR} \leq 15$. This process ensured that the performance can be quickly maximized for low SNR, while remaining accurate for signals with high SNR. For instance, 11% error (defined as the mean of the absolute values of the relative error averaged over all the test set elements and over each parameter) was obtained when trained using this scheme, with gradually decreasing SNR, and only about 21% mean error at parameter estimation was obtained on the test set when directly trained on the same range of SNRs without this scheme.

Furthermore, the classifier performed better (with an increase from 96% to 99% accuracy on the test set) when its initial weights were transferred from the fully trained predictor, i.e., the classifier was created by simply adding a *softmax* layer to the trained predictor and then trained on the dataset of signals and noise. These techniques were also useful when applying **Deep Filtering** for GW detection and characterization in real LIGO data.
2.4 Results

2.4.1 Detection

Defining sensitivity as the ratio of the number of correct detections made to the total number of inputs containing signals, at a fixed false alarm rate, the classifier achieved 100% sensitivity throughout the parameter space for signals with SNR $\geq 10$, and a single detector false alarm rate less than 0.6%. The false alarm rate of Deep Filtering can be further decreased by combining the classifications on multiple detector inputs and by computing the overlap of
Figure 2.9: **Sensitivity at SNR = 6.** The color indicates the sensitivity (%) of detection at each region of parameter space in the test set at a fixed SNR = 6 using the deeper CNN shown in Fig. 2.6. This indicates that for low BBH total mass, 1s templates may not be sufficiently long. Note that for SNR ≥ 10, however, the classifier achieved 100% sensitivity throughout the parameter space.

the template predicted by Deep Filtering with the input data to confirm each detection.

The left panel of Fig. 3.4 presents the sensitivity of detection using the shallower DNN architecture shown in Fig. 2.5. After training over the entire range of SNRs, and tuning the single detector false alarm rate to 0.6%, we found that the sensitivity of detection saturates at 100% for SNR ≥ 10, i.e., GWs with SNRs in this range are always detected. Under the same set of assumptions, i.e., training strategy and single detector false alarm rate, but now using the deeper DNN in Fig. 2.6 the right panel of Fig. 3.4 indicates that the sensitivity of detection saturates at 100% for SNR ≥ 9, and performing similarly to matched-filtering throughout the SNR range used for comparison. These results indicate that Deep Filtering can extract GW signals weaker than the background noise.

Note that Fig. 3.4 showed results averaged over the BBH parameter space under consideration. To further investigate the performance at different regions of the parameter space, Fig. 2.9 presents the sensitivity of detection, using the deeper DNN shown in Fig. 2.6 for each template in the test set assuming a fixed SNR = 6. It is worth pointing out that the sensitivity of detection for each template in the test set is 100% for SNR ≥ 10 at each region of
Figure 2.10: **Comparison with machine learning methods for detection.** The figure compares the accuracy of different machine learning methods for detection after training each with roughly 8000 elements, half of which contained noisy whitened signals with a fixed peak power, less than the background noise, and constant total mass, with the other half being pure Gaussian noise with unit standard deviation—see Section 2.7.3 for a detailed description of this comparison. An accuracy of 50% can be obtained by randomly guessing.

Parameter space. For very low mass BBH systems, at the limit of sensitivity of independent implementations of matched-filtering, i.e., SNR $\sim 6$ [Usman et al. (2016b)], the sensitivity of the classifier is relatively lower. This is because for low mass BBH GWs, the last second of the signal is contained in the high frequency regime of the aLIGO band ($\sim 4.4$kHz$/M$) beyond aLIGO’s range of optimal sensitivity. Therefore, to attain better sensitivity of detection for low mass systems, the DNNs can be trained using datasets with longer waveform templates, which may be explored in a subsequent article. On the other hand, the DNNs are capable of correctly identifying high mass BBH events. This is a promising result, because high mass BBH templates are short lived and they are difficult to accurately extract and characterize in LIGO data, as shown in [Nitz (2017)]. In summary, **Deep Filtering** performs well throughout the BBH parameter space for GW events with SNR $\geq 10$, excelling in the detection of high mass systems even at lower SNR.

To provide a baseline for comparing the classification results, we trained standard implementations of all commonly used machine learning classifiers—Random Forest, Support Vector Machine, k-Nearest Neighbors, Hidden Markov Model, Shallow Neural Networks,
Naive Bayes, and Logistic Regression—along with the DNNs on a simpler training set of 8000 elements for fixed total mass and peak signal amplitude. It can be seen that unlike DNNs, none of these algorithms were able to directly handle raw noisy data even for this simple problem as shown in Fig. 2.10.

2.4.2 Parameter estimation

Fig. 2.11 shows the variation in relative error against SNR for predicting the component masses of BBH GWs signals from the test set, embedded in Gaussian noise, for each architecture of the DNNs shown in Figs. 2.5 and 2.5. This indicates that the predictor can measure the component masses with an error of the same order as the spacing between templates for \( \text{SNR} \geq 13 \). These results show that the deeper predictor shown in Fig. 2.6 consistently outperformed matched-filtering at each SNR, as shown in the right panel of Fig. 2.11. For \( \text{SNR} \geq 50 \) both predictors could be trained to have relative error less than 5%, whereas the error with matched-filtering using the same templates was always greater than 11% with the given template bank. This means that, unlike matched-filtering, the deep learning algorithm is able to automatically perform interpolation between the known templates to predict intermediate values. Furthermore, the largest relative errors were concentrated at lower masses, because a small variation in predicted masses leads to larger relative errors in this region.

The distribution of errors and uncertainties were estimated empirically at each region of the parameter-space, and it was observed that the errors closely follow Gaussian normal distributions for each input for SNR (\( \geq 9 \)), thus allowing easier characterization of uncertainties. Fig. 2.12 presents a sample of the distribution of errors incurred in predicting the component masses of a BBH system with component masses \((57M_\odot, 33M_\odot)\). The dependence of the error with which the component masses of each template of the test dataset are recovered in each region of the parameter space is presented in Fig. 2.13 using the deeper CNN shown in Fig. 2.6 assuming a fixed SNR = 10.

Finally, we tested the baseline performance of a variety of common machine learning tech-
Figure 2.11: **Left panel: Error in parameter estimation with smaller net.** This shows the mean percentage error of estimated masses on the test sets at each SNR using the predictor DNN with 3 convolution layers shown in Fig. 2.5. Note that the DNN was trained only once over the range of SNR and was then tested at different SNR, without re-training. A mean relative error less than 20% was obtained for SNR \( \geq 8 \). At high SNR, the mean error saturates at around 11%. **Right panel: Error in parameter estimation with deeper net.** This shows the mean percentage error of estimated masses on the test sets at each SNR using the deeper CNN with 4 convolution layers shown in Fig. 2.6. A mean relative error less than 15% was obtained for SNR \( \geq 7 \). At high SNR, the mean error saturates at around 7%. Note that we were able to optimize this predictor to have less than 3% error for very high SNR (\( \geq 50 \)), which demonstrates the ability of Deep Filtering to learn patterns connecting the templates and effectively interpolate to intermediate points in parameter space.

Figure 2.12: **P-P plot of errors in parameter estimation** This is a P-P (probability) plot of the distribution of errors in predicting \( m_1 \) for test parameters \( m_1 = 57 M_\odot \) and \( m_2 = 33 M_\odot \), superimposed with different realizations of noise at SNR = 9. The best-fit is a Gaussian normal distribution with mean = 1.5\( M_\odot \) and standard deviation = 4.1\( M_\odot \). The errors followed similar Gaussian distributions in other regions of the parameter-space as well.
Figure 2.13: **Error in parameter estimation at SNR = 10.** This figure shows the mean relative error (%) in predicting the component masses for each template in the test set at a fixed SNR = 10 using the deeper CNN shown in Fig. 2.6.

Techniques including Linear Regression, k-Nearest Neighbors, Shallow Neural Networks, Gaussian Process Regression, and Random Forest on the simpler problem of predicting mass-ratio after fixing the total mass. The results shown in Fig. 2.14 indicate that, unlike DNNs, they could not predict even a single parameter accurately when trained directly on time-series data.

Having quantified the performance of Deep Filtering for GW signals emitted by non-spinning, quasi-circular BBH mergers, in the following section, the ability of the DNN-based algorithm to automatically identify new classes of signals beyond the parameter space employed for the original training and testing procedure, without retraining, is explored.

### 2.4.3 New classes of gravitational wave sources

In this section, we test the ability of Deep Filtering to detect two distinct types of signals that were not considered during the training stage, namely: (i) moderately eccentric NR simulations (approximate eccentricity $e_0 \lesssim 0.2$ when entering the aLIGO frequency band), that we recently generated with the open-source, NR software, the Einstein Toolkit Löffler et al. (2012) using the Blue Waters petascale supercomputer; and (ii) NR waveforms from
Figure 2.14: Comparison of machine learning methods for parameter estimation.
The figure shows the mean relative error obtained by various machine learning algorithms
for predicting a single parameter, i.e., mass-ratio, using a training set containing about 8000
signals with fixed amplitude = 0.6 added to white noise with unit standard deviation. Note
that scaling the alternate methods to high-dimensional parameter spaces to predict multiple
parameters is often difficult, unlike deep learning, which is more scalable, where neurons can
be added to the final layer of neural networks to predict each parameter.

the SXS catalog[Chu et al.] (2016) that describe spin-precessing, quasi-circular BBHs—each
BH having spin \( \geq 0.5 \) oriented in random directions[Chu et al.] (2016). Sample waveforms of
these GW classes as shown in Fig. 2.15. Since these NR simulations scale trivially with mass,
the data was enlarged by rescaling the signals to have different total masses. Thereafter, the
templated were whitened and added to different realizations of noise, in the same manner as
before, to produce test sets.

The DNN classifiers detected all these signals with nearly the same sensitivity as the
original test set, with 100% sensitivity for SNR \( \geq 10 \). Remarkably, the predictor quantified
the component masses of the eccentric simulations for SNR \( \geq 12 \) with a mean relative error
less than 20% for mass-ratios \( q = \{1, 2, 3, 4\} \), and less than 30% for \( q = 5.5 \) respectively. For
the spin-precessing systems that were tested, with SNR \( \geq 12 \), the mean error in predicting
the masses was less than 20% for \( q = \{1, 3\} \), respectively.

It is worth emphasizing that there exist GW algorithms that search for a wide range of
high-SNR, short-duration (burst) GW signals with minimal assumptions[Abbott et al.]
(2016a), i.e., without resorting to the use of waveform templates to identify GW events.
Figure 2.15: **New types of signals.** Left panel: This waveform was obtained from one of our NR simulations of eccentric BBH merger that has mass-ratio 5.5, total mass about 90$M_{\odot}$, and an initial eccentricity $e_0 = 0.2$ when it enters the aLIGO band. The **Deep Filtering** pipeline successfully detected this signal, even when the total mass was scaled between 50$M_{\odot}$ and 90$M_{\odot}$, with 100% sensitivity (for SNR $\geq 10$) and predicted the component masses with a mean relative error $\leq 30\%$ for SNR $\geq 12$. See also Fig. 2.18 for more types of eccentric waveforms that were used. Right panel: One of the spin-precessing waveforms obtained from the NR simulations in the SXS catalog with component masses equal to 25$M_{\odot}$ each. The individual spins are each 0.6 and oriented in un-aligned directions. The DNNs also successfully detected this signal, even when the total mass was scaled between 40$M_{\odot}$ and 100$M_{\odot}$, with 100% sensitivity for SNR $\geq 10$ and predicted the component masses with a mean relative error $\leq 20\%$ for SNR $\geq 12$. See also Fig. 2.19 for more examples of spin-precessing waveforms which were tested.

Indeed, these “burst” pipelines were used to carry out the first direct detection of GWs [Abbot et al. (2016); Abbott et al. (2016a)]. These searches do not, however, attain the same sensitivity as template-based searches for low-SNR and long-duration GW signals. Other recent advances in GW data analysis have explored the detection of spin-precessing BBH mergers using matched-filtering based algorithms [Privitera et al. (2014); Harry et al. (2016); Smith et al. (2016); Usman et al. (2016b)].

In view of the aforementioned considerations, let us discuss the importance of these findings. First of all, previous studies have reported that no matched-filtering algorithm has been developed to extract continuous GW signals from compact binaries on orbits with low to moderate values of eccentricity, and available algorithms to detect binaries on quasi-circular orbits are sub-optimal to recover these events [Tiwari et al. (2016)]. Recent analyses have also made evident that existing GW detection algorithms are not capable of accurately detecting or reconstructing the parameters of eccentric signals [Huerta et al. (2018, 2017);
However, when we scale the GW waveform from the NR simulation used on the left panel of Figure 2.15 to describe BBH mergers with mass-ratio $q = 5.5$ and total mass $M \in [50M_\odot, 90M_\odot]$, and with initial eccentricity $e_0 = 0.2$ when they enter the aLIGO band, Deep Filtering was able to identify these signals with 100% sensitivity (for SNR $\geq 10$), and recover the masses of the system with a mean relative error $\leq 30\%$ for SNR $\geq 12$. To put these results in context, the right panel of Figure 2 in Huerta et al. (2018), shows that a signal of this nature will be poorly recovered with a matched-filtering quasi-circular search.

If we now consider eccentric GW signals that are relatively weak, i.e., SNR $\geq 10$, this means that these events do not fall into the category of loud, short-duration, events that GW “burst” pipelines are able to recover without the use of templates. For reference, these low-latency GW pipelines, that use minimal assumptions, recovered short-duration high-SNR GW events such as GW150914, but missed long-duration low-SNR events, such as GW151226, which was identified by the matched-filtering based GW pipeline gstLAL [Abbott et al. (2016a)]. If we now consider that we have found similar results for a larger set of eccentric BBH signals with mass-ratios $q \leq 5.5$ and $e_0 \leq 0.2$ ten orbits before merger, then these results imply that, in the context of stationary Gaussian noise, Deep Filtering can detect and characterize eccentric BBH mergers that are poorly recovered by matched-filtering based quasi-circular searches, and whose SNRs are low enough to not be optimally recovered by GW detection pipelines with minimal assumptions. Results in the subsequent article George & Huerta (2017), show that this is also the case when real LIGO noise is used.

This ability to generalize to new categories of signals, without being shown any such examples, means that DNN-based pipelines may be able to increase the depth of existing GW detection algorithms without incurring additional computational expense. These results provide an incentive to develop DNNs that are also trained with datasets of eccentric and spin-precessing GWs to further improve the accuracy with which the Deep Filtering algorithms can detect and characterize these events in low latency.
2.4.4 Speed and computational cost

Furthermore, the simple classifier and predictor (in Fig. 2.5) are only 2MB in size each, yet they achieve excellent results. The average time taken for evaluating them per input of 1 second duration is approximately 6.7 milliseconds, and 106 microseconds using a single CPU and GPU respectively. The deeper predictor CNN (in Fig. 2.6), which is about 23MB, achieves slightly better accuracy at parameter estimation but takes about 85 milliseconds for evaluation on the CPU and 535 microseconds on the GPU, which is still orders of magnitude faster than real-time. Note that the current deep learning frameworks are not well optimized for CPU evaluation. For comparison, we estimated an evaluation time of 1.1 seconds for time-domain matched-filtering [Messick et al. (2017)] on the same CPU (using 2-cores) with the same template bank of clean signals used for training, the results are shown in Fig. 2.16. This fast inference rate indicates that real-time analysis can be carried out with a single CPU or GPU, even with DNNs that are significantly larger and trained with template banks of millions of signals. Note that CNNs can be trained on millions of inputs in a few hours using distributed training on parallel GPUs [Goyal et al. (2017)]. Furthermore, the input layer of the CNNs can be modified to consider inputs/templates of any duration, which will result in the computational cost scaling linearly with the input size. Therefore, even with 1000s inputs the analysis can still be carried out in real-time.

For applying the Deep Filtering method to a multi-detector scenario, one can directly apply the DNNs pre-trained for single detector inference separately to each detector and check for coincident detections with similar parameter estimates. Enforcing coincident detections would decrease the false alarm probability, from about 0.59% to about 0.003%. Once the Deep Filtering pipeline detects a signal then traditional matched-filtering may be applied with a select few templates around the estimated parameters to cross-validate the

\[\text{For example, a state-of-the-art CNN for image recognition} \quad \text{Ioffe & Szegedy (2015b)} \quad \text{has hundreds of layers (61MB in size) and is trained with over millions of examples to recognize thousands of different categories of objects. This CNN can process very large inputs, each having dimensions} \quad 224 \times 224 \times 3, \quad \text{using a single GPU with a mean time of} \quad 6.5 \text{milliseconds per input.}\]
Figure 2.16: **Speed-up of analysis.** The DNN-based pipeline is many orders of magnitude faster compared to matched-filtering (cross-correlation or convolution) against the same template bank of waveforms (tested on batches of inputs using both cores of an Intel Core i7-6500U CPU and an inexpensive NVIDIA GeForce GTX 1080 GPU for a fairer comparison). Note that the evaluation time of a DNN is constant regardless of the size of training data, whereas the time taken for matched-filtering is proportional to the number of templates being considered, i.e., exponentially proportional to the number of parameters. Therefore, the speed-up of Deep Filtering would be higher in practice, especially when considering larger template banks over a higher dimensional parameter space.
event and estimate confidence measure. Since only a few templates need to be used with this strategy, existing challenges to extend matched-filtering for higher dimensional GW searches may thus be overcome, allowing real-time analysis with minimal computational resources.

2.5 Discussion

It was found that the DNN architecture is resilient to the nature of the detectors’ PSD. The best-performing architecture was the same when using Gaussian noise without whitening the signals i.e., a flat PSD, and when using signals whitened with aLIGO’s design sensitivity. By incorporating examples of transient detector noise in the training set, the DNNs can also be taught to automatically ignore or classify glitches. While only simple DNNs are explored in this first study, our results show that deeper DNNs improve the accuracy of interpolation between GW templates for prediction as well as the sensitivity at low SNR, while retaining real-time performance. Even though the analysis presented in this article was carried out using Gaussian noise, the following article George & Huerta (2017) shows that the key features of this method remain the same when using real LIGO data, and that Deep Filtering is able to learn from and adapt to the characteristics of LIGO noise, without changing the architecture of the DNNs.

Deep learning is known to be highly scalable, overcoming what is known as the curse of dimensionality Goodfellow et al. (2016); Bengio & LeCun (2007). This intrinsic ability of DNNs to take advantage of large datasets is a unique feature to enable simultaneous GW searches over a higher dimensional parameter-space that is beyond the reach of existing algorithms. Furthermore, DNNs are excellent at generalizing or extrapolating to new data. Initially, we had trained a DNN to predict only the mass-ratios at a fixed total mass. Extending this to predict two component masses only required the addition of an extra neuron to the output layer. The preliminary results in this article with simulated data indicates that the DNNs may be able to detect and reconstruct the parameters of eccentric and spin-precessing
compact sources that may go unnoticed with existing aLIGO detection algorithms [Huerta et al. (2017); Tiwari et al. (2016); Huerta et al. (2014); Huerta & Brown (2013)]. The extendability of this approach to predict additional parameters such as spins, eccentricities, etc., may also be explored. Note that there are also emerging techniques to estimate and quantify uncertainties in the parameter predictions of DNNs [Perreault Levasseur et al. (2017)], which may be applied to enhance this method.

This DNN algorithm requires minimal pre-processing. In principle, aLIGO’s colored noise can be superimposed into the training set of GW templates, along with observed glitches. It has been recently found that deep CNNs are capable of automatically learning to perform band-pass filtering on raw time-series inputs [Dai et al. (2016)], and that they are excellent at suppressing highly non-stationary colored noise [Xu et al. (2015)] especially when incorporating real-time noise characteristics [Kumar & Florêncio (2016)]. This suggests that manually devised pre-processing and whitening steps may be eliminated and raw aLIGO data can be fed to DNNs. This would be particularly advantageous since it is known that Fourier transforms are the bottlenecks of aLIGO pipelines [Usman et al. (2016b)].

Once DNNs are trained with a given aLIGO PSD, they can be more quickly re-trained, via transfer learning, during a detection campaign for recalibration in real-time based on the latest characteristics of each detectors’ noise. Deep learning methods can also be immediately applied through distributed computing via citizen science campaigns such as Einstein@Home [Plitsch & Allen (2009)] as several open-source deep learning libraries, including MXNet, allow scalable distributed training and evaluation of neural networks simultaneously on heterogeneous devices, including smartphones and tablets. Low-power devices such as FPGAs and GPU chips dedicated for deep learning inference [Zhang et al. (2015); Han et al. (2016)] may even be placed on the GW detectors to reduce data transfer issues and latency in analysis.

DNNs automatically extract and compress information by finding patterns within the training data, creating a dimensionally reduced model [Hinton & Salakhutdinov (2006)].
fully trained DNNs are each only 2MB (or 23MB for the deeper model) in size yet encodes all the relevant information from about 2500 GW templates (about 200MB, before the addition of noise) used to generate the training data. Once trained, analyzing a second of data takes only milliseconds with a single CPU and microseconds with a GPU. This means that real-time GW searches could be carried out by anyone with an average laptop computer or even a smartphone, while big datasets can be processed rapidly in bulk with inexpensive hardware and software optimized for inference. The speed, power efficiency, and portability of DNNs could allow rapidly analyzing the continuous stream of data from GW detectors [George & Huerta (2017)] or other astronomical facilities.

**Scope for Improvements**

One may construct a multi-dimensional template bank based on state-of-the-art semi-analytical models and all available NR-based waveforms. Thereafter, one can superimpose samples of real aLIGO noise (and non-Gaussian transients) on these templates and carry out an intensive training procedure. Once this process is finished, the DNN may be used for real-time classification and parameter estimation while being periodically re-trained with more gravitational waveforms and recent aLIGO noise.

Out-of-core training techniques may be needed if the training data is too large to fit in RAM. The size of the data can be reduced by figuring out more optimal placements of templates within the parameter-space. Nonetheless, the DNNs have been observed to perform monotonically better when trained with increasing amounts of data [Goodfellow et al. (2016)]. An alternative would be to incorporate noise addition into the initial layers of the DNN so that the training set is smaller, containing only the clean signal templates, and new noise will be superimposed automatically in each iteration.

Our DNNs contain only 15 hidden layers in total. Developing efficient architectures of DNNs, that can be trained quickly and evaluated faster, is an active area of research. For example, GoogLeNet [Szegedy et al. (2015)] is a very deep neural network that is an order of
magnitude smaller in size and evaluates inputs much faster than competitors while achieving the same or higher levels of accuracy. Residual nets or ResNets [He et al., 2015] are the current state-of-the-art models for image processing and have been shown to monotonically improve when adding more layers.

CNNs are limited by the fact that they can only use fixed length tensors as inputs and outputs. On the other hand, Recurrent Neural Networks (RNNs), the deepest of all neural networks, have cyclic internal structures and are well-suited for time-series analysis since they can make decisions based on a stream of inputs rather than a vector of fixed length [Schmidhuber, 2015]. A powerful type of RNN called LSTM for Long-Short-Term-Memory (discovered in 1997 by Hochreiter and Schmidhuber [Hochreiter & Schmidhuber, 1997]) is capable of remembering long-term dependencies in the input sequence. This suggests that there is a large scope for improvement in developing efficient architectures of DNNs for scientific data analysis.

Since the architecture of the classifier and predictor is almost identical, it may be possible to fuse their initial layers to minimize computational costs. Furthermore, hierarchical multi-task learning is possible with DNNs [Caruana, 1993; Zeng & Ji, 2015], thus allowing a single network to classify inputs into categories and sub-categories, while also performing parameter estimation for each type of signal.

Stacking chunks of time-series data to produce multi-dimensional tensors can facilitate processing massive quantities of data efficiently on modern hardware, for example, to find signals that are very long in duration like inspirals of neutron stars. Experimenting with different loss functions (e.g. mean squared relative error) may improve the accuracy in certain regions of the parameter-space. The accuracy of the DNNs can be further enhanced by training an ensemble of different models and taking an average of the results for each input [Goodfellow et al., 2016].

Due to the fast computation speed, low power consumption, and portability, it will be far more efficient to perform real-time analysis by directly sending a continuous stream of
data to dedicated GPU inference engines at the aLIGO site to maximize the detection rate and minimize lags due to data transfer. FPGAs are another promising development that can further accelerate DNNs [Zhang et al. (2015)]. Having powerful deep-learning-optimized machines at each detector will make it possible to continuously re-train the DNNs with the latest aLIGO data thus recalibrating them in real-time based on the characteristics of the current noise. Anticipating the growth of AI, many chip manufacturers are heavily investing in developing specialized hardware for training and evaluating DNNs.

aLIGO uses a variety of independent sensors to monitor the environment and assess data quality. There are independent algorithms to estimate periods which must be vetoed due to disturbances that lead to a loss in detector sensitivity. It may be possible to train DNNs to perform this process of vetoing by taking into account simultaneous inputs from all sensors to determine data quality.

Stretches of real aLIGO data may be used to re-train the DNNs by adding injections of all available templates of gravitational waveforms from binary systems, including inspirals of BNSs and stellar mass BBHs using catalogs of numerical relativity simulations covering spin-precessing and eccentric binaries. The classifier can be trained to identify new categories of glitches and other non-Gaussian sources of noise, that closely mimic signals, in conjunction with unsupervised learning techniques for anomaly detection.

2.6 Conclusion

The framework for signal processing presented in this article may be applied to enhance existing low-latency (online) GW data analysis techniques in terms of both performance and scalability and could help in enabling real-time multimessenger astrophysics observations in the future. Deep CNNs were exposed to time-series template banks of GWs, and allowed to develop their own strategies to detect and predict source parameters for a variety of GW signals embedded in highly noisy simulated data. The DNN-based method introduced in
this article has been applied in George & Huerta (2017) to build a Deep Filtering pipeline, trained with real LIGO noise, including glitches, which detected true GWs in real LIGO data and accurately estimated their parameters. These results, provide an incentive to further improve and extend Deep Filtering to target a larger class of GW sources, incorporating glitch classification and clustering George et al. (2018), and GW denoising Shen et al. (2017) algorithms to accelerate and broaden the scope of GW searches with aLIGO and future GW missions.

It was found that even though the DNNs were trained using a dataset of GWs that describe only quasi-circular, non-spinning BBH mergers, Deep Filtering is capable of detecting and characterizing low-SNR GW signals that describe non-spinning, eccentric BBH mergers, and quasi-circular, spin-precessing BBH mergers. This provides motivation to enhance the Deep Filtering algorithm introduced herein to predict more parameters by including millions of spin-precessing and eccentric templates for training potentially using distributed computing methods in HPC facilities.

Employing DNNs for multimessenger astrophysics offers opportunities to harness AI computing with rapidly emerging hardware architectures and software optimized for deep learning. In addition, the use of state-of-the-art HPC facilities will continue to be used to numerically model GW sources, getting insights into the physical processes that lead to EM signatures, while also providing the means to continue using distributed computing to train DNNs.

This new approach may help in enabling real-time multimessenger observations by providing immediate alerts for follow-up after GW events. Since deep CNNs also excel at image processing, they have been applied for transient identification in large sky surveys and high cadence surveys, respectively Cabrera-Vives et al. (2017). These results, combined with the analysis presented here and in George & Huerta (2017) suggest extensive scope for deep learning techniques to develop a new framework to further the multimessenger astrophysics program.
Acknowledgments

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. The eccentric numerical relativity simulations used in this article were generated on Blue Waters with the open source, community software, the Einstein Toolkit. We express our gratitude to Gabrielle Allen, Ed Seidel, Roland Haas, Miguel Holgado, Haris Markakis, Justin Schive, Zhizhen Zhao, other members of the NCSA Gravity Group, and Prannoy Mupparaju for their comments and interactions and to the many others who provided feedback. We thank Vlad Kindratenko for granting us unrestricted access to numerous GPUs and HPC resources in the Innovative Systems Lab at NCSA. We are grateful to NVIDIA for their generous donation of several Tesla P100 GPUs, which was used in this analysis. We also acknowledge Wolfram Research for technical assistance and for developing the software stack used to carry out this study and draft this publication.

2.7 Appendix

The Wolfram Language (Mathematica) was used for training and testing the Deep Neural Networks (DNNs) and comparator methods as well as for data processing and visualization. Detailed documentation for all the functions mentioned in this section can be found at https://reference.wolfram.com

2.7.1 Preparing training and testing data

We generated inspiral-merger-ringdown gravitational wave (GW) templates that describe binary black hole (BBHs) systems, with zero component-spins, on quasi-circular orbits with
the open source effective-one-body (EOB) model\cite{Taracchini2014,Bohe2017}, that is implemented in LIGO’s Algorithm Library (LAL), which is also used currently to generate template banks for aLIGO analysis pipelines. For the training set, we chose component masses from $5.75M_\odot$ to $75M_\odot$ in steps of $1M_\odot$ such that $m_1 > m_2$. The test set contained intermediate masses, i.e., masses from $5.25M_\odot$ to $75M_\odot$ in steps of $1M_\odot$. The validation set contained intermediate masses, i.e., masses from $5M_\odot$ to $75M_\odot$ in steps of $1M_\odot$. We deleted points having mass-ratio greater than 10. Each of these masses were rounded so that the mass-ratio was a multiple of 0.1. This gave the distribution shown in Fig. 2.4.

The EOB waveforms were generated from an initial GW frequency of 15Hz using a sampling rate of 8192Hz. For this study, we used the dominant waveform mode $(\ell, m) = (2, 2)$. The detectors’ strain is given by \cite{Sathyaprakash2009}: $h(t) = h_+(t)F_+ + h_\times(t)F_\times$, where $F_+ , F_\times$ represent the antenna pattern of the detectors. As we had assumed optimally oriented systems, which satisfy $F_+ = 1$, $F_\times = 0$, only the $h_+$ component was extracted from these templates.

We selected the final 1 second of data from each of these waveforms and re-sampled them at 8192Hz. Next, they were all whitened by dividing with aLIGO’s design sensitivity amplitude spectral density of noise, in Fourier space. We used the “Zero-detuned High Power” sensitivity of aLIGO, shown in Fig. 2.3.

Each template is a vector of 8192 real numbers labeled with the component masses. Random examples in the final training template bank is shown in Fig. 2.17. Before every training session, each template was independently translated to the left by up to 0.2 seconds randomly, and padded with zeros on the right to keep the total length invariant, to produce multiple time-series with the same parameters so that the positions of the peaks are not always at a fixed location. The mean position of the signal peak after translations was at 0.8s. Batches of different realizations of Gaussian noise with standard deviation set according to the desired SNR was added to each of these templates, and the resulting time-series were scaled to have zero mean and unit standard deviation, before each session. Note that the addition of noise
Figure 2.17: **Examples of training templates.** This shows 28 randomly chosen examples of clean signal templates in our training dataset, obtained with the EOB code, after whitening with the aLIGO PSD but prior to addition of noise. The original test sets contained the same type of signals with different component masses. These signals are all produced by mergers of non-spinning, non-eccentric BBHs.
may instead be incorporated into the training at run-time and changed automatically in each round, to make the process more efficient.

2.7.2 Designing and training neural networks

For training both of our DNNs, the back-propagation algorithm was performed over multiple rounds, known as epochs, until the errors were minimized. Stochastic gradient descent with mini-batches [Ruder (2016)] has been the traditional method used for back-propagation. This technique uses an estimate of the gradient of the error over subsets of the training data in each iteration to change the weights of the DNN. The magnitude of these changes is determined by the “learning rate”. Variations of this with adaptive learning rates such as ADAM (Adaptive Momentum Estimation) have been shown to achieve better results more quickly [Kingma & Ba (2014)], therefore we chose this method as our learning algorithm.

We employed a random trial-and-error procedure for optimizing the hyperparameters, in which different values of hyperparameters such as stride, depth, kernel size, and dilation, were manually tuned for each layer and the performance of each DNN was interactively monitored during training as shown in Fig. 2.7. We did not use any zero padding for the convolution and pooling layers, since the sampling rate was high enough so that points near the edges were irrelevant. We experimented with the ramp (ReLU) and tanh functions for the nonlinear activation layers and found that the ReLU performed the best, which is typically the case for convolutional networks [Krizhevsky et al. (2012)]. A reshape layer was added at the input in order to convert vector inputs into a matrix with a single row which can be processed by the convolution layers designed for image processing. These DNNs were designed with the NetChain function and trained with the NetTrain function in the Wolfram Language. This neural network functionality was internally implemented via the open-source MXNet deep learning library [Chen et al. (2015)] written in C++, which uses standard well-established methods for training. The source code is available at [https://github.com/dmlc/mxnet](https://github.com/dmlc/mxnet).

When training, the TargetDevice was set to “GPU”. The initial learning rate was set
to 0.001 and weights of each neuron was automatically initialized randomly according to the Xavier (Glorot) method \cite{Glorot2010} (and manually reset when needed with NetInitialize). The standard ADAM method was used, with the parameters $\beta_1 = 0.93$ and $\beta_2 = 0.999$ which are the exponential decay rates for the first and second moment estimates respectively. L2 regularization was set to zero. The size of the mini-batches was chosen automatically depending on the specifications of the GPU and datasets. The loss functions were selected to be the mean squared error for prediction and cross-entropy loss for the detection/classification task. The maximum number of overall batches was set to 100000, however, the training was often stopped earlier manually when over-fitting was found to occur, i.e., error on the validation set stopped decreasing. Most of the intensive training was done on an NVIDIA Tesla P100 GPUs with version 11 of the Wolfram Language, however, a few test sessions were performed with NVIDIA Tesla K40, GTX 1080, and GT 940M GPU.

For all sessions, the SNR for each time-series was randomly sampled from the range 5-10 and multiplied with a constant factor. Initially this constant factor was set to 20, which implies that the first session of training had SNR $\geq 100$. Then this constant factor in subsequent training rounds was lowered in decreasing step sizes until it was 1, i.e., the final SNR range was uniformly sampled between 5 and 15. For prediction, each time-series was labeled with the component masses of the BBH system that generated the original waveforms.

For classification, we initially added batches of noise having half this size, and desired SNR, to the clean templates and appended pure noise to get the same number of elements in total for each session. The labels were changed to “True” or “False”, depending on whether a signal is present, for training the classifier. The weights of the trained predictor was extracted and used to initialize the same layers in the classifier. We initialized this classifier with the pre-trained weights of the predictor and added a softmax layer to produce probabilities of different classes as output. The NetDecoder function was used within the classifier to convert the numeric vectors of probabilities to classes with labels “True” or “False”. Then we trained
this network using the same procedure with \( \text{SNR} \geq 100 \) and and slowly decreased the SNR every round until a final SNR distribution uniformly sampled in the range 5 to 10. The fraction of noise in the training set was tuned by trial and error to about 87.5\% to lower the false alarm rate to the desired value.

Considering that all the DNNs we tested are tiny by modern standards and only a small space of hyperparameters was explored by us, we expect that higher accuracies over a wider range of parameters and types of signals can be obtained by exploring more complex configurations of DNNs, choosing more optimal hyperparameters, and using a larger set of carefully-placed training templates covering the full range of GW signals.

### 2.7.3 Comparisons with other methods

Comparisons with other machine learning methods used built-in standard implementations of these common algorithms as documented in the Wolfram Language. Open-source versions of these methods are also available in libraries such as scikit-learn. Optimal parameters for each model were chosen automatically by the Classify and Predict functions. The time-series of 1 second duration sampled at 8192Hz was directly used as inputs to all methods. The mean of the absolute values of relative error on the test set was measured for prediction. For classification, the accuracy on the test set was measured using the ClassifierMeasurements function. The steps followed are described below. To provide a fair comparison, each method was directly given the same raw time-series as inputs. Note that it may be possible to improve the performance of any machine learning method by providing hand-extracted “expert” features instead or a DNN may be used as a feature extractor for each of the alternative methods.

For comparison with different methods for prediction (parameter estimation), we used the same EOB waveforms as before but fixed the total mass to be \( 60M_\odot \) for training and testing, to predict only the mass-ratio. Thus, we used 91 templates covering mass-ratios from 1 to 10 in steps of 0.1 for training and 15 templates with intermediate mass-ratios
for testing. The size of this training data was enhanced by adding different realizations of
gaussian noise, scaled by the same total mass of $60M_\odot$ and labeled with the mass-ratio. Then
88 different realizations of noise were added to each of the training templates to produce a
total of 8008 time-series for training and 264 different realizations of noise were added to
each of the testing templates to obtain 3960 time-series for testing. A validation set of 2640
elements was also produced by adding another 176 different realizations of noise to each of
the testing templates. The noise was chosen to have a Gaussian distribution and a unit
standard deviation. The amplitude of all the signals were set to 0.6 and added to the noise
to create the inputs for training and testing. The inputs were then normalized to have unit
standard deviation and zero mean. Smaller datasets were used because the other methods
are not implemented efficiently on GPUs, unlike DNNs, therefore the training procedure
was done on a high-performance CPU machine over several days. The predictor DNN was
initialized randomly and re-trained with this new dataset for comparison. The mean relative
errors we obtained with the different methods, shown in Fig. 2.14, are as follows: DNN -
10.92%, Shallow Neural Networks - 49.93%, Gaussian Process Regression - 67.43%, Linear
Regression - 67.50%, Random Forest - 67.59%, k-Nearest Neighbors - 51.18%.

For comparing the classifiers, we trained all the methods from scratch with the same
set of templates, labeled “True”, appended with 50% pure Gaussian noise labeled “False”,
comprising of 7662 time-series. The testing and validation sets contained 3516 elements,
each having 50% noise. The ratio of the amplitude of the signals to the standard deviation
of the noise was fixed at 0.6. The classifier DNN was also initialized randomly and re-
trained with this dataset. Note that this implies that DNNs can be successfully trained
with much smaller datasets for the detection task alone at fixed high SNR. Larger number
of templates were used in our analysis in order to perform parameter estimation, which is a
harder problem than classification since the parameter-space is continuous as opposed to a
finite discrete set of classes, and to improve the performance at low SNR. The accuracy of
these methods obtained on the test set, shown in Fig. 2.14, are as follows: DNN - 99.81%

To measure the speed of evaluation (inference) of the DNNs on new inputs, the `AbsoluteTiming` function was used to measure the total time for the evaluation of each method over batches of 1000 inputs and the average time per input was computed. The benchmarks were all run with Mathematica 11, which uses the Intel MKL library, on a Windows 10 64-bit machine with an Intel Skylake Core i7-6500U CPU. A desktop-grade NVIDIA GTX 1080 GPU was used for the measuring the speed-up of analysis with DNNs, instead of the expensive high-performance GPUs that were used for training, since this has a price closer to a desktop CPU and thus provides a fair comparison against the performance of the CPU at similar costs. The measured times averaged over batches of inputs were 6.67 milliseconds and 106 microseconds per input (vector of length 8192) with the CPU and GPU respectively.

We used a standard implementation of time-domain matched-filtering (similar to [Messick et al. (2017)](https://example.com)) with the same template bank of clean signals by computing the cross-correlation (which was same as convolution with time-reversed templates) of an input of 1 second duration from the test set against the same templates in the training set using the same sampling rate. The parameters were estimated to be those of the best matching template. The threshold of single-detector matched-filter SNR required for detection was tuned to be about 5.3 to have a false alarm rate similar to the classifier’s. Since the peak of the signal was shifted within 0.2 seconds while training the DNNs, we also assumed the same window for the location of the peak for matched-filtering by truncating the templates to have 0.8 second duration (removing the part near the edges, which does not contain the signal). The `ListCorrelate` function, which uses the Intel MKL library, was used to perform this computation and the mean time per input on the same CPU (optimized to use both cores) over 1000 parallelized runs was measured to be about 1.1 seconds.
For timing the larger DNN deployed for the Image Identification Project\footnote{https://www.imageidentify.com/}, we used the NetModel function to obtained the pre-trained model in version 11 of the Wolfram Language. This DNN is based on the Inception V2 model originally proposed in Ioffe & Szegedy\cite{IoffeSzegedy2015}. The timing of inference was done using an NVIDIA Tesla P100 GPU on a batch of 1000 inputs, each being an RGB image having a resolution of $224 \times 224$ pixels. The average time per input for batches of 1000 images was measured to be 6.54 milliseconds.

### 2.7.4 Measuring accuracy and errors

For computing the sensitivity at each SNR, we applied the DNN classifier to time-series inputs containing the true signals, produced by adding 10 different realizations of noise to each of the clean templates (about 2500) in the test set and computed the ratio of detected signals to the total number of inputs. The SNR was varied from 2 to 17 in steps of 0.5. Therefore, about 0.8 million seconds of data, in total, sampled at 8192Hz was used for constructing the sensitivity plots. The false alarm rate was measured by applying the classifier to 100,000 realizations of Gaussian noise with duration 1s and sampling rate 8192Hz.

For measuring the mean relative errors in prediction at each SNR, we applied the predictor on time-series inputs produced by adding 10 different realizations of noise to each of the clean templates in the test set and averaged this at each SNR. The absolute value of the relative errors in predicting each component mass was averaged. The SNR was varied from 2 to 17 in steps of 0.5. Thus, about 0.8 million seconds of data in total, sampled at 8192Hz, was also used for preparing each of these plots.

The distribution of errors for randomly chosen templates in the test set were measured after the addition of 1000 different realizations of noise to each of them at fixed SNR. We verified that the errors closely match Gaussian distributions using standard probability-probability (P-P) plots at randomly chosen points in the parameter-space for SNR $\geq 9$. For lower SNR, the distribution was slightly skewed. The best fitting parameters of the nor-
normal distribution was automatically chosen by the ProbabilityPlot function and a random sample is shown in Fig. 2.12.

Although this analysis was originally intended for quasi-circular, non-spinning binaries, we tested the performance of the DNNs on new classes of signals without extra training. The eccentric NR signals used in this study were generated using the open source software, the Einstein Toolkit [Löffler et al. (2012)], on the Blue Waters supercomputer. For reproducibility purposes, we are including the metadata information of the simulations we used as auxiliary supplementary material. A large catalog of eccentric NR simulations will be presented in a subsequent publication. The waveforms extracted from the Einstein Toolkit data are rendered in natural units of \( M \), and describes BBH systems with a total mass of \( 1M_\odot \). All 4 waveforms we used for this article are also attached and have the identifiers E0001, E0009, E0017, E0025, and L0020 for mass-ratios 1, 2, 3, 4, and 5.5 respectively. The first four simulations had an eccentricity of 0.1 and the last had 0.2 when entering the LIGO band. The parameter files that we used for our eccentric simulations were modified versions of the open-source parameter file [Wardell et al. (2016)]. We had used resolutions of 32, 36, and 40 grid points across each BH matching resolutions used in typical production simulations. We verified that these exhibited strong convergent behavior. Full simulation data will be provided upon request.

As discussed before, since GW templates scale trivially with mass, more templates were produced by scaling the eccentric NR waveforms to have total masses between \( 85M_\odot \) and a maximum mass depending on the mass-ratio, to ensure that the simulations provided enough data for about 1 second and the component masses lies between \( 5M_\odot \) and \( 75M_\odot \), which is the range of component masses that the predictor was originally trained for. The maximum mass for each mass-ratio was set so that the largest component mass was \( 75M_\odot \). We again used the real (+) component of the dominant \((\ell, m) = (2, 2)\) mode. A random template at each mass ratio is shown in Fig. 2.18.

The mean relative errors were predicted on a test set obtained after adding 10000 different
Figure 2.18: **Examples of eccentric signals.** These are the 5 simulations of eccentric BBH systems with different mass-ratios, which we used to test the DNNs. Each of these signals were scaled to have different total masses by stretching in time to enlarge the size of the test set. They were produced with the open-source Einstein Toolkit software on the Blue Waters supercomputer. The initial conditions were chosen such that the eccentricity was 0.1 for the first 4 simulations and 0.2 for the final simulation for each system as it enters the aLIGO band.
realizations of noise with SNR = 10 and 12 for every value of total mass for each mass-ratio. The mean of the absolute values of relative error was calculated. We separately analyzed the prediction rates for each signal, since they differed by a large rate for different mass-ratios. A step size of $0.5M_\odot$ was used to vary the total mass by stretching E0001, E0009, E0017, E0025, and L0020. The range used for total masses were different for different mass ratios according to the constrain that individual masses should lie between $5M_\odot$ and $75M_\odot$. For measuring sensitivity of detection, we used the combined dataset of all these templates used for prediction, at fixed SNR of 10, each added to 10000 realizations of noise to create a single test set.

For testing with spin-precessing systems, we used waveforms extracted from 4 NR simulations that describe quasi-circular, spin-precessing BBH systems obtained from the publicly available catalog of simulations performed by the SXS collaboration [Chu et al. (2016)], hosted at https://www.black-holes.org/waveforms/catalog.php. Full data and parameters of each simulation can be found at this website. The BBH configurations we selected, labeled SXS:BBH:0050, SXS:BBH:0053, SXS:BBH:0161, SXS:BBH:0163, represent compact binaries with the largest values of spin (larger than 0.5 each) oriented in arbitrary directions, so as to exacerbate the effect of spin-precession, and serves as strong tests of the robustness of the detection and parameter reconstruction algorithms. Their mass-ratios were 3, 3, 1, and 1 respectively.

The spin-precessing NR waveforms we selected correspond to the highest quality waveforms for each simulation. This was found in the highest resolution runs (labeled with highest “Lev”). The second order extrapolation to infinite radius (N2-Extrapolated file) within the “rhOverM_Asymptotic_GeometricUnits.h5” files was selected. Since we assumed optimally oriented systems for this study, we chose + component of the dominant waveform mode, $(\ell, m) = (2, 2)$, which captures the signatures of spin-precession. The total mass was again scaled in the same manner, with the constraints on the range of component masses and that the signal should last 1 second. These NR simulations were longer than the eccentric
Figure 2.19: Examples of spin-precessing signals. These are the 4 GW simulations of spin-precessing BBHs from the SXS catalog, which we used to test the DNNs. Each of these signals were also scaled to have different total masses by stretching in time to enlarge the size of the test set. The individual spins of each system was higher than 0.5, and the orientation was in arbitrary directions, i.e., the spins were not aligned or anti-aligned. Full details of these simulations are available at [https://www.black-holes.org/waveforms/catalog.php](https://www.black-holes.org/waveforms/catalog.php)
ones, therefore, the lower limit of total mass was set to $60M_\odot$. The upper limit was chosen such that, for each mass-ratio, the largest component mass was $75M_\odot$. A randomly chosen template for each system is shown in Fig. 2.19.

The mean of the absolute value of relative errors in predicting component masses was computed separately for the different mass-ratios (1 and 3), in the same manner, for spin-pressing systems. For each signal, 10000 sets of different noise realizations were added at each value of total mass, which was varied in steps of $0.5M_\odot$. The sensitivity of detection was measured on the combined set of signals obtained in the same manner as before.
Chapter 3

Application of Deep Learning with Real Advanced LIGO Data

The recent Nobel-prize-winning detections of gravitational waves from merging black holes and the subsequent detection of the collision of two neutron stars in coincidence with electromagnetic observations have inaugurated a new era of multimessenger astrophysics. To enhance the scope of this emergent field of science, we had pioneered the use of deep learning with convolutional neural networks, that take time-series inputs, for rapid detection and characterization of gravitational wave signals. This approach, Deep Filtering, was initially demonstrated using simulated LIGO noise. In this chapter, we present the extension of Deep Filtering using real data from LIGO, for both detection and parameter estimation of gravitational waves from binary black hole mergers using continuous data streams from multiple LIGO detectors. We demonstrate for the first time that machine learning can detect and estimate the true parameters of real events observed by LIGO. Our results show that Deep Filtering achieves similar sensitivities and lower errors compared to matched-filtering while being far more computationally efficient and more resilient to glitches, allowing real-time processing of weak time-series signals in non-stationary non-Gaussian noise with minimal resources, and also enables the detection of new classes of gravitational wave sources that may go unnoticed with existing detection algorithms. This unified framework for data analysis is ideally suited to enable coincident detection campaigns of gravitational waves and their multimessenger counterparts in real-time.

3.1 Introduction

The first detection (GW150914) of gravitational waves (GWs), from the merger of two black holes (BHs), with the advanced Laser Interferometer Gravitational-wave Observatory (LIGO) (The LIGO Scientific Collaboration et al., 2015) has set in motion a scientific revolution (Abbott et al., 2016) leading to the Nobel prize in Physics in 2017. This and subsequent groundbreaking discoveries (Abbott et al., 2016a, 2017b,a) were brought to fruition by a trans-disciplinary research program at the interface of experimental and theoretical physics, computer science and engineering as well as the exploitation of high-performance computing (HPC) for numerical relativity simulations (Einstein, 1915; Abbott et al., 2016b; Usman et al., 2016b) and high-throughput computing facilities for data analysis (Huerta et al., 2017a; Weitzel et al., 2017).

Matched-filtering, the most sensitive GW detection algorithm used by LIGO, currently targets a 3D parameter space (compact binary sources with spin-aligned components on quasi-circular orbits) (Gerosa et al., 2013; Rodriguez et al., 2016, 2015)—a subset of the 8D parameter space available to GW detectors (Antonini et al., 2016; Naoz et al., 2013; Samsing et al., 2014; Huerta et al., 2017; Lehner & Pretorius, 2014). Recent studies also indicate that these searches may miss GWs generated by compact binary populations formed in dense stellar environments (Klimenko et al., 2016; Huerta et al., 2017; Huerta et al., 2014; Huerta & Brown, 2013). Extending these template-matching searches to target spin-precessing, quasi-circular or eccentric BBHs is computationally prohibitive (Harry et al., 2016).

Based on the aforementioned considerations, we need a new paradigm to overcome the limitations and computational challenges of existing GW detection algorithms. An ideal candidate would be the rapidly advancing field called Deep Learning, which is a highly scalable machine learning technique that can learn directly from raw data, without any manual feature engineering, by using deep hierarchical layers of “artificial neurons”, called neural networks, in combination with optimization techniques based on back-propagation.
and gradient descent (Lecun et al., 2015; Goodfellow et al., 2016). Deep learning, especially with the aid of GPU computing, has recently achieved immense success in both commercial applications and artificial intelligence (AI) research (Lecun et al., 2015; Schmidhuber, 2015; Esteva et al., 2017; Silver et al., 2017; Moravčík et al., 2017; van den Oord et al., 2016; O’Shea et al., 2016) and has also been applied in astrophysics (Hezaveh et al., 2017; Hinners et al., 2017; Sedaghat & Mahabal, 2017; Pearson et al., 2017; Caron et al., 2017; George et al., 2018; Zevin et al., 2016; Bahaadini et al., 2017).

Our technique, Deep Filtering (George & Huerta, 2018), employs a system of two deep convolution neural networks (CNNs (LeCun et al., 1998a)) that directly take time-series inputs for both classification and regression. In our foundational article (George & Huerta, 2018), we provided a comprehensive introduction to the fundamental concepts of deep learning and CNNs along with a detailed description of this method. Our previous results showed that CNNs can outperform traditional machine learning methods, reaching sensitivities comparable to matched-filtering for directly processing highly noisy time-series data streams to detect weak GW signals and estimate the parameters of their source in real-time, using GW signals injected into simulated LIGO noise.

In this article, we extend Deep Filtering to analyze GW signals in real LIGO noise. We demonstrate, for the first time, that Deep Learning can be used for both signal detection and multiple-parameter estimation directly from extremely weak time-series signals embedded in highly non-Gaussian and non-stationary noise, once trained with some templates of the expected signals. Our results show that deep CNNs achieve performance comparable to matched-filtering methods, while being several orders of magnitude faster and far more resilient to transient noise artifacts such as glitches. We also illustrate how Deep Filtering can deal with data streams of arbitrary length from multiple detectors. Most importantly, this article shows for the very first time that machine learning can successfully detect and recover true parameters of real GW signals observed by LIGO. Furthermore, we show that after a single training process, Deep Filtering can automatically generalize to noise having...
new Power Spectral Densities (PSDs) from different LIGO events, without re-training.

Our results indicate that Deep Filtering can interpolate between elements in the template bank, generalize to several new classes of signals beyond the training data, and, surprisingly, detect GW signals and measure their parameters even when they are contaminated by glitches. We present experiments demonstrating the robustness of Deep Filtering in the presence of glitches, which indicate its applicability in the future for glitch classification and clustering efforts essential for GW detector characterization (Zevin et al., 2016). Deep learning, in principle, can learn characteristics of noise in the LIGO detectors and develop better strategies than matched-filtering, which is known to be optimal only for Gaussian noise. Since all the intensive computation is diverted to the one-time training stage of the CNNs, template banks of practically any size may be used for training after which continuous data streams can be analyzed in real-time with a single CPU, while very intensive searches can be rapidly carried out using a single GPU. The initial estimates provided by Deep Filtering may be used to instantly narrow down the parameter space of GW detections, which can then be followed up with existing pipelines using a few templates around the predicted parameters to obtain more informative parameter estimates and confidence measurements, thus accelerating GW analysis with minimal computational resources across the full parameter space of signals.

3.2 Methods

Deep Filtering consists of two steps, involving a classifier CNN and a predictor CNN, with similar architectures, as described in our previous article (George & Huerta, 2018). The classifier has an additional softmax layer which returns probabilities for True or False depending on whether a signal is present. The classifier is first applied to the continuous data stream via a sliding window. If the classifier returns higher probability for True, the predictor is applied to the same input to determine the parameters of the source. In a multi-
detector scenario, the **Deep Filtering** CNNs may be applied separately to each data stream and the coincidence of detections with similar parameters would strengthen the confidence of a true detection, which can then be verified quickly by matched-filtering with the predicted templates.

In this work, we have used injections of GW templates originating from quasi-circular, non-spinning, stellar-mass BBH systems, which LIGO/Virgo is expected to detect with the highest rate \cite{Belczynski2016}. We assumed the source is optimally oriented with respect to the detectors which reduces our parameter-space to the two individual masses of the BBH system, which we restricted in the range $5M_\odot$ to $75M_\odot$ such that their mass-ratios were between 1 and 10. We fixed the input duration to 1 second, and a sampling rate of 8192Hz, which is more than sufficient for the events we are considering. These are arbitrary choices, as the input size of the CNNs can be easily modified to take inputs with any duration or sampling rate from any number of detectors.

The datasets of waveform templates used to train and test our CNNs were obtained using the open-source, effective-one-body (EOB) code \cite{Taracchini2014}. Our training set contained about 2500 templates, with BBHs component masses sampled in the range $5M_\odot$ to $75M_\odot$ in steps of $1M_\odot$. The testing dataset also contained approximately 2500 templates with intermediate component masses separated from the training set by $0.5M_\odot$ each. Subsequently, we produced copies of each signal by shifting the location of their peaks randomly within the final 0.2 seconds to make the CNNs more resilient to time translations. This means that in practice, our algorithm will be applied to the continuous data stream using a 1-second sliding window with offsets of 0.2 seconds.

We obtained real LIGO data from the LIGO Open Science Center (LOSC) around the first 3 GW events, namely, GW150914, LVT151012, and GW151226. Each event contained 4096 seconds of real data from each detector. We used noise sampled from GW151226 and LVT151012 for training and validation of our model and noise from GW150914 was used for testing. These tests ensure that our method is able to generalize to different noise
Figure 3.1: **Sample signal injected into real LIGO noise.** The red time-series is an example of the input to our Deep Filtering algorithm. It contains a hidden BBH GW signal (blue) from our test set which was superimposed in real LIGO noise from the test set and whitened. For this injection, the optimal matched-filter SNR = 7.5 (peak power of this signal is 0.65 times the power of background noise). The component masses of the merging BHs are $57M_\odot$ and $33M_\odot$. The presence of this signal was detected directly from the (red) time-series input with over 99% sensitivity and the source’s parameters were estimated with a mean relative error less than 10%.

distributions, also in the presence of transient glitches, since it is well known that the PSD of LIGO is highly non-stationary, varying widely with time. Therefore, if Deep Filtering performs well on these test sets, it would also perform well on data from future time periods, without being re-trained.

Next, we superimposed different realizations of noise randomly sampled from the training set of real LIGO noise from the two events GW151226 and LVT151012 and injected signals over multiple iterations, thus amplifying the size of the training datasets. The power of the noise was adjusted according to the desired optimal matched-filter Signal-to-Noise Ratio (SNR (Owen & Sathyaprakash 1999)) for each training round. The inputs were then whitened with the average PSD of the real noise measured at that time-period. We also scaled and mixed different samples of LIGO noise together to artificially produce more training data and various levels of Gaussian noise was also added to augment the training process. However, the testing results were measured using only pure LIGO noise not used in training with true GW signals or with signals injected from the unaltered test sets (see Fig. 3.1).
Figure 3.2: **Spectrograms of real LIGO noise test samples.** We used signals injected into real data from the LIGO detectors in this article, ensuring that the training and testing sets did not contain noise from the same events. These are some random examples of real glitches that were present in our test set of LIGO noise. The **Deep Filtering** method takes the 1D strain directly as input and is able to correctly classify glitches as noise and detect true GW signals as well as simulated GW signals injected into these highly non-stationary non-Gaussian data streams, with similar sensitivity compared to matched-filtering.

We used similar hyperparameters to our original CNNs [George & Huerta 2018] with a slightly deeper architecture. There were 4 convolution layers with the filter sizes to 64, 128, 256, and 512 respectively and 2 fully connected layers with sizes 128 and 64. The standard ReLU activation function, $\max(0, x)$, was used throughout as the non-linearity between layers. We used kernel sizes of 16, 16, 16, and 32 for the convolutional layers and 4 for all the (max) pooling layers. Stride was chosen to be 1 for all the convolution layers and 4 for all the pooling layers. We observed that using dilations [Yu & Koltun 2016] of 1, 2, 2, and 2 in the corresponding convolution layers improved the performance. The final layout of our predictor CNN is shown in Fig. 3.3.

We had originally optimized our CNN architectures to deal with only Gaussian noise having a flat PSD. However, we later found that this model also obtained the best performance with noise having the colored PSD of LIGO, among all the models we tested. This indicates that our CNNs will be robust to a wide range of noise distributions. Furthermore, pre-training the CNNs on Gaussian noise (transfer learning) before fine-tuning on the limited amount of real noise prevented over-fitting, i.e., memorizing only the training data without
Figure 3.3: **Architecture of deep convolutional neural network.** This is the dilated 1D CNN used as the predictor which outputs the component masses of the BBH system. The classifier has the same architecture, except for a softmax layer added at the end which outputs the probability for the presence of a GW signal. The input is a time-series vector of length 8192 corresponding to 1s of data sampled at 8192Hz. The classifier is applied separately to continuous data streams from each detector using a sliding window. If the classifier detects a signal in coincidence across multiple detectors, then the inputs are fed to the predictor which estimates the parameters of the GW source. The classifier is separated from the predictor and softmax regression is used so that neurons can be added to the final layer of the classifier for representing different types of signals and glitches and specialized predictors may be designed to be applied depending on the class of the signal.
generalizing to new inputs. We used the Wolfram Language (Mathematica) neural network functionality, built using the open-source MXNet framework (Chen et al., 2015), that uses the cuDNN library (Chetlur et al., 2014) for accelerating the training on NVIDIA GPUs. The learning algorithm was ADAM (Kingma & Ba, 2014) and other details were the same as before (George & Huerta, 2018).

While training, we used the curriculum learning strategy in our first article (George & Huerta, 2018) to improve the performance and reduce training times of the CNNs while retaining performance at very high SNR. By starting off training inputs having high SNR ($\geq 100$) and then gradually increasing the noise in each subsequent training session until a final SNR distributed in the range 4 to 15, we found that the performance of prediction can be quickly maximized for low SNR while retaining performance at high SNR. We first trained the predictor on the datasets labeled with the BBH masses and then copied the weights of this network to initialize the classifier and then trained it on datasets having 90% pure random noise inputs, after adding a softmax layer. This transfer learning procedure, similar to multi-task learning, decreases the training time for the classifier and improves its sensitivity.

3.3 Results

The sensitivity (probability of detecting a true signal) of the classifier as a function of SNR is shown in Fig. 3.4. We achieved 100% sensitivity when SNR is greater than 10. The false alarm rate was tuned to be less than 1%, i.e., 1 per 100 seconds of noise in our test set was classified as signals. Given independent noise from multiple detectors, this implies our 2-detector false alarm rate would be less than 0.01%, when the classifier is applied independently to each detector and coincidence is enforced. Although the false alarm rate can be further decreased by tuning the fraction of noise used for training or by checking that the predicted parameters are consistent, this may not be necessary since running matched-filtering
Figure 3.4: **Sensitivity of detection with real LIGO noise.** The curve shows the sensitivity of detecting GW signals injected in real LIGO noise (from LOSC) from our test set using **Deep Filtering** and with matched filtering with the same template bank used for training. Note that the SNR is on average proportional to $10 \pm 1.5$ times the ratio of the amplitude of the signal to the standard deviation of the noise for our test set. This implies that we are capable of detecting signals significantly weaker than the background noise. While matched-filtering has the advantage of being optimized with the PSD of the LIGO noise in the test set, **Deep Filtering** was only trained on noise from other events, therefore our results demonstrate the ability of the CNNs to automatically generalize to non-stationary LIGO noise having different PSDs without retraining.

Figure 3.5: **Error in parameter estimation with real LIGO noise.** This shows the mean percentage absolute error of estimating masses on our testing signals at each SNR, injected in real LIGO noise from events not used for training, compared to matched filtering using the same template bank that was used for training. While the mean error of matched-filtering, with the same template bank used for training, is always greater than 11% at every SNR we can see that the **Deep Filtering** method is able to interpolate to test set signals with intermediate parameter values.
pipelines with a few templates close to our predicted parameters can quickly eliminate these false alarms.

Our predictor was able to successfully measure the component masses given noisy GW signals, that were not used for training, with an error lower than the spacing between templates for optimal matched-filter SNR \( \geq 15.0 \). The variation in relative error against SNR is shown in Fig. 3.5. We observed that the errors follow a Gaussian distribution for each region of the parameter space for SNR greater than 10. For high SNR, our predictor achieved mean relative error less than 10\%, whereas matched-filtering with the same template bank always has error greater than 10\%. This implies that Deep Filtering is capable of interpolating between templates seen in the training data.

Although, we trained only on simulated quasi-circular non-spinning GW injections, we applied Deep Filtering to the LIGO data streams containing a true GW signal, GW150914, using a sliding window of 1s width with offsets of 0.2s through the data around each event from each detector. This signal was correctly identified by the classifier at the true position in time and each of the predicted component masses were within the published error bars (Abbott et al., 2016). There were zero false alarms after enforcing the constraint that the detection should be made simultaneously in multiple detectors. This shows that deep learning is able to generalize to real GW signals after being trained only with simulated GW templates injected into LIGO noise from other events with different PSDs. A demo showing the application of Deep Filtering to GW150914 can be found here: tiny.cc/CNN.

The data from the first LIGO event, that was used for testing, contained a large number of non-Gaussian transient noise called glitches. Some of these can be seen in Fig. 3.2. Therefore, our results demonstrate that the Deep Filtering method can automatically recognize these glitches and classify them as noise. This suggests that by adding additional neurons for each “glitch” class, Deep Filtering could serve as an alternative to glitch classification algorithms based on two-dimensional CNNs applied to spectrograms of LIGO (Zevin et al., 2016; George et al., 2018) or machine learning methods based on manually engineered...
Figure 3.6: **Examples of sine-Gaussian glitches.** These are some samples of simulated sine-Gaussian glitches from our test set. We found that our classifier was able to correctly differentiate GW signals from these glitches and classify them as noise when they were injected into real LIGO data streams. This suggests that Deep Filtering can be extended to create a unified pipeline for glitch classification along with signal detection and parameter estimation.

Furthermore, we conducted some experiments to show the resilience of Deep Filtering to transient disturbances, with a simulated set of sine-Gaussian glitches, which cover a broad range of morphologies found in real LIGO glitches, following (Powell et al., 2017) (see Fig. 3.6 for some examples). We ensured that a different set of frequencies, amplitudes, peak positions, and widths were used for training and testing. We then injected some of these glitches into the training process and found that the classifier CNN was able to easily distinguish new glitches from true signals, with a false alarm rate less than 1%, using only single detector inputs. When we applied the standard naive matched-filtering algorithm to the same test set of glitches, approximately 30% of glitches were classified as signals due to their high SNR. This is because matched-filtering is unable to distinguish signals from loud glitches having similar frequencies. Note that additional signal consistency tests and coherence across detectors can be enforced to decrease this false alarm rate for both methods.

We then tested the performance of Deep Filtering, when a signal happens to occur in coincidence with a glitch, i.e., the signal is superimposed with both a glitch and real LIGO noise (see Fig. 3.7). We trained the CNNs by injecting glitches from the training set and
measured the sensitivity of the classifier on the test set signals superimposed with glitches sampled from the test set of glitches. We found that over 80% of the signals with SNR of 10 were detected, and their parameters estimated with less than 30% relative error, even after they were superimposed with glitches. These results are very promising, motivating an in-depth investigation, since we may be able to automatically detect GW signals that occur during periods of bad data quality in the detectors using Deep Filtering, whereas currently such periods are often vetoed and left out of the analysis by LIGO pipelines.

Another important experiment that we carried out was to inject waveforms obtained from simulations of eccentric BBH systems, with eccentricities between 0.1 and 0.2 when entering the LIGO band that we performed using the open-source Einstein Toolkit (Löffler et al., 2012) on Blue Waters, as well as waveforms from spin-precessing binaries from the public SXS catalog (Chu et al., 2016) as described in (George & Huerta, 2018). We found that, after injecting these signals into real LIGO noise from the test set events, they were detected by our classifier with the same sensitivity as the original test set of quasi-circular BBH waveforms, with 100% sensitivity of detection and less than 35% error in estimating masses for SNR ≥ 10, thus demonstrating its ability to automatically generalize to GW signals from new classes of BBH mergers beyond the training data to some extent. This is particularly promising since
recent studies indicate that moderately eccentric BBH populations may be missed by quasi-
circular GW searches [Huerta et al. 2017; Tiwari et al. 2016; Huerta et al. 2014, 2017b]. Parameter estimation on these new classes of signals could be more challenging, without training, as spin and eccentricity may have the effect of mimicking features of systems of different masses. We expect that by training on the full parameter space of signals, we will be able to decrease this error.

Both our CNNs are only 23MB in size each, yet encodes all the relevant information from about 2,500 GW templates (~300MB) of templates and several GB of noise used to generate the training data. The time-domain matched-filtering algorithm used for comparison required over 2s to analyze 1s inputs on our CPU. The average time taken for evaluating each of our CNNs per second of data is approximately 85 milliseconds and 540 microseconds using a single CPU core and GPU respectively, thus enabling analysis even faster than real-time. While the computational cost of matched-filtering grows exponentially with the number of parameters, the Deep Filtering algorithm requires only a one-time training process, after which the analysis can be performed in constant time. Therefore, we expect the speed-up compared to matched-filtering to further increase by several orders of magnitude when the analysis is extended to larger number of parameters. When considering the full range of signals that span a very high-dimensional parameter space which cannot be sampled densely due to computational costs, we expect Deep Filtering may have higher sensitivity due to its ability to interpolate and because the one-time training process can be carried out with template banks much larger than what is feasible to use with matched-filtering.

3.4 Conclusion

In this article, we have shown for the first time that CNNs can be used for both detection and parameter estimation of GW signals in LIGO data. This new paradigm for real-time GW analysis may overcome outstanding challenges regarding the extension of established GW
detection algorithms to higher dimensions for targeting a deeper parameter space of astrophysically motivated GW sources. The results of Deep Filtering can be quickly verified, and the time of coalescence computed, via matched-filtering with a small set of templates in the region of parameter space predicted by the CNN. Therefore, by combining Deep Filtering with well-established GW detection algorithms we may be able to accelerate multimessenger campaigns, pushing the frontiers of GW astrophysics and fully realize its potential for scientific discovery.

The intrinsic scalability of deep learning can overcome the curse of dimensionality (Goodfellow et al., 2016) and take advantage of massive datasets (Najafabadi et al., 2015). This ability could enable fast automated GW searches covering millions or billions of templates over the full range of parameter-space that is beyond the reach of existing algorithms. Extending Deep Filtering to predict any number of parameters such as spins, eccentricities, etc., or additional classes of signals or noise, is as simple as adding an additional neuron for each new parameter, or class, to the final layer and training with noisy waveforms with the corresponding labels. Furthermore, the input dimensions of the CNNs can be enlarged to take time-series inputs from multiple detectors, thus allowing coherent searches and measurements of parameters such as sky locations.

The results presented in this article provide a strong incentive to extend Deep Filtering to cover the parameter space of spin-aligned BBHs on quasi-circular orbits and beyond. This study is underway, and will be described in a subsequent article. In addition to our primary results, we have also presented experiments exhibiting the remarkable resilience of this method for detection in periods of bad data quality, even when GW signals are contaminated with non-Gaussian transients. This motivates including additional classes of real glitches, e.g., from the Gravity Spy project (Zevin et al. 2016), to the training process to automatically classify and cluster glitches directly from the time-series inputs. It may be possible that by investing more effort in hyperparameter tuning and by experimenting with deeper architectures we may be able to outperform the sensitivity of matched-filtering
searches in the future (since matched-filtering is proven to be the optimal algorithm only for Gaussian noise). Therefore, a single, robust, and efficient data analysis pipeline for GW detectors, based on Deep Filtering, that unifies detection and parameter estimation along with glitch classification and clustering in real-time with very low computational overhead may be built and deployed in the next observing run of LIGO in the near future.

Deep Filtering can be extended to provide instant alerts with accurate parameter estimates for EM follow-up campaigns. Furthermore, the initial point estimates provided by the Deep Filtering predictor may be used to inform and accelerate computationally intensive offline Bayesian parameter estimation methods (Veitch et al., 2015b; Pankow et al., 2015) to estimate the full posterior distributions for each parameter by constraining the parameter space of new GW detections. It is also possible to apply CNNs to obtain uncertainty distributions for the parameters (Levasseur et al., 2017), which will be explored in a following article. As deep CNNs excel at image processing, applying the same approach to analyze raw telescope data may also accelerate the subsequent search for transient EM counterparts.

Our results also suggest that, given templates of expected signals, Deep Filtering can be used as a generic tool for efficiently detecting and estimating properties of highly noisy time-domain signals embedded in Gaussian noise or non-stationary non-Gaussian noise, even in the presence of transient disturbances. Therefore, we anticipate that the techniques we have developed for analyzing weak signals hidden in complex noise backgrounds could be useful in many other domains of science and technology.

3.5 Acknowledgements

This research is part of the Blue Waters sustained-petascale computing project, supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. The eccentric numerical rel-
ativity simulations used in this article were generated with the open source, community software, the Einstein Toolkit on the Blue Waters petascale supercomputer and XSEDE (TG-PHY160053). We express our gratitude to Gabrielle Allen, Ed Seidel, Roland Haas, Miguel Holgado, Haris Markakis, Zhizhen Zhao and other members of the NCSA Gravity Group along with Prannoy Mupparaju for comments and interactions and to the many others who reviewed our manuscript, including Jin Li and Kai Staats. We acknowledge the LIGO/Virgo collaboration, especially the CBC and MLA groups, for their feedback and for use of computational resources including the NVIDIA DGX-1 machine. We thank Vlad Kindratenko for granting us access to GPUs and HPC resources in the Innovative Systems Lab at NCSA. We are grateful to Wolfram Research for providing technical assistance and Mathematica licenses and to NVIDIA for donating several Tesla P100 GPUs, which we used for our analysis.
Chapter 4

Anomaly Classification and Unsupervised Clustering with Deep Transfer Learning

Gravitational wave detection requires a detailed understanding of the response of the LIGO and Virgo detectors to true signals in the presence of environmental and instrumental noise. Of particular interest is the study of anomalous non-Gaussian transients, such as glitches, since their occurrence rate in LIGO and Virgo data can obscure or even mimic true gravitational wave signals. Therefore, successfully identifying and excising these anomalies from gravitational wave data is of utmost importance for the detection and characterization of true signals, and to accurately compute their significance. To facilitate this work, we present the first application of Deep Learning combined with Transfer Learning to show that knowledge from pretrained models for real-world object recognition can be transferred for classifying spectrograms of glitches. To showcase this new method, we use a dataset of twenty-two classes of glitches, curated and labeled by the Gravity Spy project using data collected during LIGO’s first discovery campaign. We demonstrate that our Deep Transfer Method enables an optimal use of very deep convolutional neural networks for glitch classification given small and unbalanced training datasets, significantly reduces the training time, and achieves state-of-the-art accuracy above 98.8%, lowering the previous error rate by over 60%. More importantly, once trained via transfer learning on the known classes, we show that our neural networks can be truncated and used as feature extractors for unsupervised clustering to automatically group together new unknown classes of glitches and anomalous signals. This novel capability is of paramount importance to identify and remove new types of glitches.
which will occur as the LIGO/Virgo detectors gradually attain design sensitivity.

4.1 Introduction

The advanced Laser Interferometer Gravitational-wave Observatory (LIGO) (The LIGO Scientific Collaboration et al., 2015) and the European advanced Virgo (Acernese et al., 2015) detectors are the largest and most sensitive interferometric detectors ever built. The cutting-edge design and fabrication of LIGO’s subsystems enable the sensing of changes in LIGO’s arm-length thousands of times smaller than the diameter of a proton (The LIGO Scientific Collaboration et al., 2015; The LIGO Scientific Collaboration & The Virgo Collaboration, 2016). These instruments have already detected multiple gravitational wave (GW) signals produced from mergers of black holes (Abbott et al., 2016a,a, 2017b,a) as well as neutron stars (Abbott et al., 2017b), which was accompanied by electromagnetic radiation across the entire spectrum (Abbott et al., 2017). As LIGO/Virgo gradually attain design sensitivity, they will transition from their current discovery mode into an astronomical observatory that will routinely detect new GW sources, providing insights into astrophysical objects, and processes that cannot be observed through any other means (LIGO Scientific Collaboration et al., 2013; Sathyaprakash & Schutz, 2009).

For LIGO/Virgo to realize their full potential for scientific discovery, it is necessary to ensure that their sensing capabilities are not hindered by unwanted non-Gaussian noise transients, known as glitches, which contaminate GW data. There is extensive ongoing work in the gravitational wave community on separating these glitches from signals, and classifying them based on their characteristics, which is a non-trivial task requiring “intelligent” algorithms given that noise transients vary widely in duration, frequency range and morphology, spanning a wide parameter space that is challenging to model accurately (Cornish & Littenberg, 2015; Powell et al., 2015, 2017; Mukund et al., 2017; Blackburn et al., 2008). Furthermore, since the LIGO/Virgo detectors are undergoing commissioning between each
observing run, we expect that new types of glitches will be identified as they attain reach sensitivity (The LIGO Scientific Collaboration & The Virgo Collaboration, 2016; The LIGO Scientific Collaboration & the Virgo Collaboration, 2016; The LIGO Scientific Collaboration & Abbott, 2016).

Accurately classifying glitches is essential for several reasons: a) This will prevent false GW detections due to coincident glitches across multiple detectors that closely mimic signals. This issue is expected to occur once LIGO/Virgo operate close to design sensitivity, since the rate of detectable GW signals will increase, and the overlap with transient noise will increase for minute-long scale GW signals. b) Rapidly identifying and excising glitches will enhance detector sensitivity, and enable a better quantification of the significance of GW signals that are contaminated by glitches. c) The LIGO/Virgo detectors have thousands of instrumental and environmental channels to monitor changes that occur due to environmental or hardware issues. By carefully tracking down these changes, we aim to identify their source and eliminate them promptly to ensure that the detectors’ output data is still suitable for GW data analysis. This would increase the duty cycle of GW detectors enabling more observations having scientific impact.

The complex and time-evolving nature of glitches makes them an ideal case study to apply machine learning algorithms—methods that learn from examples rather than being explicitly programmed. Machine learning techniques are becoming essential since, as the number of GW detectors increase along with their up-time, the amount of glitches would rise dramatically making it impossible to classify them all manually. Machine learning can be divided into supervised and unsupervised learning depending on whether labeled, structured data is used for training the algorithms (Goodfellow et al., 2016). Deep learning, i.e., machine learning based on deep artificial neural networks (Lecun et al., 2015; Goodfellow et al., 2016), is one of the fastest growing fields of artificial intelligence research today, having outperformed competing methods in many areas of machine learning applications, e.g., image classification, face recognition, natural language understanding and translation,
speech recognition and synthesis, game-playing (e.g., Go, Poker), and self-driving vehicles. Therefore, deep learning algorithms are also expected to have excellent performance for the characterization of gravitational wave detectors. In this article, we focus on deep learning with Convolutional Neural Networks (CNNs), the leading approach for computer visions tasks (Lecun et al., 2015), using spectrograms computed from the time-series glitches as inputs.

Recent efforts on this front include Gravity Spy, an innovative interdisciplinary project that provides an infrastructure for citizen scientists to label datasets of glitches from LIGO via crowd-sourcing (Zevin et al., 2016). Supervised classification algorithms based on this dataset were presented in the first Gravity Spy article (Zevin et al., 2016) and were further discussed in Ref. (Bahaadini et al., 2017). In the latter study, deep multi-view models were introduced to enhance classification accuracy using multiple-duration spectrograms of each glitch stitched-together to form a larger input image. These algorithms employed deep learning CNN models which were 4 layers deep, and achieved overall accuracies close to 97% for glitch classification. However, it was found that glitch classes with very few labeled samples were more difficult to classify with the same level of accuracy.

As new classes of glitches are uncovered in the near future, the design and hyperparameters of these CNNs will have to be modified accordingly to distinguish subtle features between a larger set of classes. Considering these issues, one may be interested in exploring the use of very deep CNNs designed for the harder task of classifying thousands of categories of real-world objects. However, the small size of the Gravity Spy dataset, compared to the datasets used to train these state-of-the-art CNNs for real-world object recognition, would lead to poor results due to immediate overfitting, i.e., memorization of features in the training data without generalizing to the testing data. Therefore, previous attempts at glitch classification from spectrograms have used relatively shallow CNNs, which were designed and

---

1Hyperparameters refer to several quantities that have to be manually chosen to determine the architecture of the neural networks (e.g., overall design, number of layers, sizes of the convolutions, depth, padding, types of layers and activations, etc.)
trained from the ground-up (Zevin et al., 2016; Bahaadini et al., 2017). With this approach, lower input image resolution is used to keep model size manageable while significant time and effort is needed to optimize the architecture (hyperparameters) of the CNNs and to train them from scratch. To directly employ deep CNN models with well-established architectures optimized for computer vision, for this glitch classification problem, a different approach is necessary.

In this article, we present Deep Transfer Learning, a new method for glitch classification that leverages the complex structure and deep abstraction of pre-trained state-of-the-art CNNs used for object recognition, and fine-tunes them throughout all layers to accurately classify glitches after re-training on a small dataset of LIGO spectrograms. We show that this technique achieves state-of-the-art results for glitch classification with the Gravity Spy dataset, attaining above 98.8% overall accuracy and perfect precision-recall on 8 out of 22 classes, while significantly reducing the training time to a few minutes. Furthermore, new types of glitches can be classified accurately given very few labeled examples with this technique. We also demonstrate that features learned from real-world images by very deep CNNs are directly transferable for the classification of spectrograms of time-series data from GW detectors, and possibly also for spectrograms in general, although the two datasets are very dissimilar. The CNNs we use were originally designed for over 1000 classes of objects in ImageNet. Therefore, our algorithm can be easily extended to classify hundreds of new classes of glitches in the future, especially since this transfer learning approach requires only a few labeled examples of a new class. Furthermore, we briefly outline how new classes of glitches can be automatically found and grouped together as they occur in the data stream by using our trained CNNs as feature extractors for unsupervised or semi-supervised clustering algorithms. This has been an area of intense research for LIGO/Virgo instrumentalists (Aasi et al., 2015; Powell et al., 2015, 2017). The work we introduce in this article represents a significant step in this direction.

This article is organized as follows. Section 4.2 describes the methods and datasets used
for applying transfer learning to develop accurate CNNs for glitch classification. Section 4.3 summarizes our results. We discuss immediate applications of our method and its extension for finding new classes of glitches in an unsupervised manner in Section 4.4. Conclusions and a brief description of future strategies to assist ongoing efforts for LIGO data analysis are provided in Section 4.5.

4.2 Methods

In this Section we provide a succinct description of neural networks and transfer learning. Thereafter, we present our methods and procedure for applying transfer learning in the context of glitch classification.

4.2.1 Neural Networks

CNNs are currently the best performing method for image classification, object recognition, and a variety of other image processing tasks (Lecun et al., 2015). They serve as excellent feature extractors by allowing end-to-end learning of features and representations from raw image data for classification, thus eliminating the need for feature engineering, i.e., hand-extraction of features or representations by human experts. CNN-based algorithms have recently achieved super-human results in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) (Deng et al., 2009a), and has been outperforming all other methods since 2012.

State-of-the-art CNNs for real-world object recognition are extremely powerful, and thus training them from the ground up on a small dataset would lead to poor performance. This is because the capacity of the model, i.e., the degrees of freedom (number of parameters) of the model would be much larger than necessary to fit the small data distribution, thus resulting in what is known as “overfitting” (see section 5.2 in the Deep Learning book (Goodfellow et al., 2016)), where the algorithm memorizes the training samples without generalizing to
the test set. The symptoms of overfitting can be alleviated by using regularization tech-
niques, such as dropout (Srivastava et al., 2014) and by limiting the number of training
iterations (early stopping). However, in practice, an extremely large labeled dataset (typi-
cally a million samples) is required to train these randomly initialized deep CNNs for best
performance (Goodfellow et al., 2016).

The Gravity Spy crowd-sourcing project mobilizes citizen scientists to hand-label spec-
trograms obtained from LIGO time-series data. The rationale for this approach is that
humans are capable of distinguishing different classes of noise from spectrograms after being
shown only a few examples, which indicates that generic pattern recognizers developed in
humans for real-world object recognition are also useful when distinguishing spectrograms
of glitches. Therefore, this motivated us to apply a similar approach where a CNN is first
trained on a large database of labeled real-world objects, and this knowledge is subsequently
transferred by re-training (fine-tuning) the CNN on the dataset of spectrograms. In machine
learning literature, this idea is referred to as transfer learning, which we describe in the
following Section.

### 4.2.2 Transfer Learning

Transfer learning is an essential ingredient for artificial intelligence, where knowledge learned
in one domain for some task can be transferred to another domain for a different task (Yosin-
ski et al., 2014). In the context of deep learning, transfer learning is commonly implemented
by pre-training a deep neural network, e.g., a CNN, on a large labeled dataset followed by
fine-tuning (continued training) on a different dataset of interest, which is usually smaller.
This approach has been successfully applied in many areas of computer vision (Sharif Raza-
vian et al., 2014).

It is well known that the initial layers of a CNN always learn to extract simple generic
features (e.g., edges, corners, curves, color blobs, etc.), which are applicable to all types of
images, whereas the final layers represent highly abstract and data-specific features (Sharif
Therefore, using a model fully-trained on a large dataset and then fine-tuning it on a different dataset is expected to result in higher accuracy and a faster training process, compared to training the same CNNs from scratch, due to the shared features present in the initial layers.

To demonstrate the power of transfer learning for classifying glitches, we compare the performance of the most popular CNN models for object recognition, namely Inception (Christian et al., 2014) version 2 and 3, ResNet (Kaiming et al., 2015), and VGG (Karen & Andrew, 2014), all of which were leading entries in recent ILSVRC competitions. These CNNs were pre-trained on a large dataset of images — i.e., ImageNet (Deng et al., 2009b), which contains 1.2 million labeled images of real-world objects belonging to 1000 categories (see 4.13 for sample images) — over the course of 2 to 3 weeks using multiple GPUs by other research groups. We obtained the open-source weights from these models, and used them to initialize the CNNs, before fine-tuning each model on the dataset of glitches.

We show that transfer learning allows us to apply these powerful CNNs for classifying glitches using a very small training dataset of spectrograms to obtain significantly higher accuracies, reduce the training time by several orders of magnitude, and eliminate the need for optimizing hyper-parameters. Furthermore, we show how information from multiple duration spectrograms can be efficiently encoded into a single image in different color channels to enhance information provided to these CNN models.

### 4.2.3 Dataset

The Gravity Spy dataset, from the first observing run of LIGO, contains labeled spectrogram samples from 22 classes of glitches shown in 4.1. Within the dataset, each sample contains 4 images recorded with durations: 0.5s, 1.0s, 2.0s, and 4.0s. These images were hand-labeled by citizen scientists participating in the Gravity Spy project, and the accuracy of the labeling was greatly enhanced by cross-validation techniques within the Gravity Spy infrastructure, also involving experts from the LIGO Detector Characterization team.
Figure 4.1: Classes of glitches in the Gravity Spy dataset from the first observing run of LIGO. The No_Glitch class is omitted in the figure. For each glitch sample, spectrograms with durations of 0.5s, 1.0s, 2.0s, and 4.0s are available. The objective is to predict the classes given the images.

These glitch classification efforts has already led to resolving the origin of a few classes of glitches such as "Air_Compressor" and "Power_Line".

We cropped each image closely around its frame, removing axis labels and white-spaces. Subsequently, the resolution of these images were $481 \times 575 \times 3$, with the third dimension corresponding to the RGB color channels. For training and testing, these images were down-sampled to the optimal resolution for each CNN model, either $224 \times 224 \times 3$ or $299 \times 299 \times 3$.

We used the recent version of the Gravity Spy dataset as of April 19, 2017, which has been slightly modified since the previous publications \cite{Zevin2016, Bahaadini2017}, with two additional classes, namely 1080_Lines and 1400_Ripples, and about 300 extra elements in the Violin_Mode class. We randomly split this dataset, containing about 8500 elements, into two parts such that approximately 80% of samples in each class was in the training set, and 20% of each class was in the testing set.

With this transfer learning approach, we did not need to engage in any kind of hyperparameter optimization that could have led to overfitting on the test set. We directly used the same hyperparameters as the pre-trained models designed for object recognition. For
cross-verification of the performance, the Inception models were trained independently from the rest of the models, using a different deep learning software framework and different randomized test-train splits of the data.

For training VGG16, VGG19, and ResNet50, we only used spectrogram images with 2.0s duration, since they contained sufficient information to resolve most of the long and short duration glitches. Since the dataset is small, image augmentation was used to artificially enhance the size of the dataset (except when training the Inception models). For each input image, we subtracted the mean of the training samples, then randomly shifted it by 0 to 10 pixels in horizontal and vertical directions, and randomly zoomed in and out with factors up to 0.05 $\times$. The mean image of all the training samples was also subtracted from each sample used for testing.

For the Inception models, we found that we could improve the accuracy, without any image augmentation, by encoding multiple-duration spectrograms of each glitch into each of the RGB color channels to present maximum information to the CNNs. The 1.0s, 2.0s, and 4.0s duration spectrograms were converted to gray-scale, and then merged to produce RGB images with each channel encoding the different durations. Examples of these encoded images are shown in 4.12. Although to the human eye, this looks like a single merged RGB image, the CNNs simply see this as a set of 3 matrices (tensor) and can easily separate the channels and analyze them independently. Therefore, in the course of training, the network learns to associate the correct time-scales for the corresponding channels and thus benefits from having zoomed-in and zoomed-out views of the same spectrogram image. While previous work had used concatenated or parallel-views of multiple-duration spectrograms as inputs to specially designed CNNs trained from scratch (Bahaadini et al. 2017), our encoding method has the advantage of allowing the direct use of pre-trained CNN architectures designed for ImageNet for classification of multiple-duration spectrograms.
4.2.4 Training

The final fully-connected layer in each CNN model was replaced with another fully-connected layer having 22 neurons corresponding to each glitch class. To reduce overfitting, a problem that the neural network achieves a perfect classification accuracy on training data but compensates the accuracy on testing data, we added several dropout layers (Srivastava et al., 2014) (which randomly turn neurons off to prevent co-adaptation between neurons and thus improves their ability to generalize) at the beginning of the network and right before the output layers. Apart from this, we also added one additional layer after the first trainable convolutional layer for ResNet. The softmax function is used as the final layer in each model to provide probabilities of each class as the outputs.

When training each model, we utilized the standard cross entropy loss (cost) function due to its good performance in classification problems in combination with the softmax layer (Goodfellow et al., 2016). We fine-tuned across all the layers since the dataset of glitches is very different from the objects in the ImageNet data. Each model was trained up to 100 epochs with the usual iterative procedure (Goodfellow et al., 2016) and check-pointed after each epoch. The best performing check-points were chosen subsequently.

We used the standard AdaGrad (John et al., 2011) and ADAM (Kingma & Ba, 2014) optimizers as the learning algorithm (Goodfellow et al., 2016) for training. Specifically, we used AdaGrad for ResNet, VGG16 and VGG19 with initial learning rate set to $10^{-4}$ and $\epsilon = 10^{-6}$ and the ADAM method with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the Inception models. Inception V2 and V3 models were implemented in the Wolfram Language, which internally uses the open-source MXNet (Chen et al., 2015) framework for training. VGG and ResNet models were implemented and trained independently on a different randomized train-test split using TensorFlow (Abadi et al., 2016) via Keras (Chollet et al., 2015).
4.3 Results

As with all machine learning research, we have provided empirical measurements on the independent test set which was not used to train the neural network. Therefore the accuracy of the model on the test set serves as a good measure of the performance of the model on new inputs. A naive implementation of a logistic regression classifier on top of features extracted from the spectrograms using Inception models pre-trained on ImageNet, produced over 95% accuracy without fine-tuning any layers. This strongly indicated that features learned for real-world object recognition are also directly useful for classification of spectrograms of LIGO noise transients, and thus offered incentive to now train (fine-tune) every layer. The results after fine-tuning are presented below.

For the CNNs implemented in this paper, both InceptionV2 and InceptionV3 achieved over 98% accuracy in fewer than 10 epochs of training (less than 20 minutes), VGG16 and VGG19 achieved over 98% accuracy within 30 epochs of training. We compare the results of these CNNs trained with the transfer learning method with the CNN models trained from scratch [Zevin et al., 2016; Bahaadini et al., 2017] in Table 4.1. We found that each of our models consistently achieved over 98% accuracy for many epochs, thus indicating that the performance is robust, regardless of the stopping criteria, and therefore the model is not overfitting on the test set.

The precision and recall obtained with each model on every class is reported in Table 4.2. Where precision refers to the ratio of correctly predicted samples of a class (true positive) to samples predicted to this class (true positive + false positive), and recall refers to the ratio of correctly predicted samples of a class (true positive) to samples labeled with this class (true positive + false negative). Explicit equations for both terms is defined in Equation 4.1 and 4.2. With InceptionV3, we achieved perfect precision and recall on 8 classes: 1080Lines, 1400Ripples, Air_Compressor, Chirp, Helix, Paired_Doves, Power_Line, and Scratchy. With ResNet50, we achieved perfect precision and recall on 7 classes: 1080Lines,
1400Ripples, ExtremelyLoud, Helix, PairedDoves, Scratchy, and ViolinMode.

\[
\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (4.1)
\]

\[
\text{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \quad (4.2)
\]

We have shown the confusion matrices for ResNet50 and InceptionV3 in 4.2 and 4.3, respectively. Confusion matrices of the remaining models are included in the supplementary materials. The accuracies of each of these models, and previously published CNN models trained from scratch on the same dataset are reported in 4.1. Note that each of our models consistently under-performed with less than 98% accuracy when trained with random initializations, i.e., without applying transfer learning. Therefore, it is evident that the networks trained with the transfer learning approach achieve better results than the CNN models trained from the ground up on the same dataset.

Both ResNet50 and InceptionV3 achieved the highest accuracy of 98.84% on the test set despite being trained independently via different methods on different splits of the data with 2.0s scans and RGB encoded scans respectively. ResNet50 and InceptionV3 also obtained 100.00% accuracy when considering the top-5 predictions, which implies that given any input, the true class can be narrowed down to within 5 classes with 100.00% confidence. This means that for each input from the test set, the true class is always within the set of 5 classes with the highest probabilities as predicted by the network. This is particularly useful, since the true class of a glitch is often ambiguous to even human experts.

4.4 Discussion

The classification of glitches is of paramount importance to characterize GW detectors. The results presented in 4.1 indicate that transfer learning with state-of-the-art image classifica-
Figure 4.2: Confusion matrix of ResNet50 after fine-tuning on the dataset of glitches. The accuracy is 98.84%. Note that ResNet is the current state-of-the-art CNN model for a variety of image processing tasks including object recognition.
Figure 4.3: Confusion matrix for InceptionV3. The accuracy is also 98.84%. It can be seen that different errors were made compared to ResNet50. Note that the Chirp glitch class (which is also the shape for true GW signals from mergers of binaries) and the Paired Doves class (which only had 24 training examples) was identified with perfect precision and recall.
Table 4.1: Accuracy on the Test Set

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Top-1(20 classes)</th>
<th>Top-1</th>
<th>Top-2</th>
<th>Top-3</th>
<th>Top-4</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuned-InceptionV3</td>
<td>98.76%</td>
<td>98.84%</td>
<td>99.71%</td>
<td>99.88%</td>
<td>99.94%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Tuned-InceptionV2</td>
<td>98.70%</td>
<td>98.78%</td>
<td>99.59%</td>
<td>99.71%</td>
<td>99.94%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Tuned-ResNet50</td>
<td>98.76%</td>
<td>98.84%</td>
<td>99.71%</td>
<td>99.83%</td>
<td>99.94%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Tuned-VGG16</td>
<td>98.02%</td>
<td>98.15%</td>
<td>99.36%</td>
<td>99.71%</td>
<td>99.83%</td>
<td>99.88%</td>
</tr>
<tr>
<td>Tuned-VGG19</td>
<td>98.08%</td>
<td>98.21%</td>
<td>99.31%</td>
<td>99.60%</td>
<td>99.71%</td>
<td>99.71%</td>
</tr>
<tr>
<td>Previous Best</td>
<td>96.72%</td>
<td>96.70%</td>
<td>98.32%</td>
<td>99.13%</td>
<td>99.31%</td>
<td>99.36%</td>
</tr>
</tbody>
</table>

The table lists the top-1 to top-5 accuracies for different CNNs on the testing set. The top-1 accuracy for 20 classes (excluding 1080Lines and 1400Ripples) is also shown for comparison with previous publications. We re-trained the 4 layer merged-view CNN model described in the publications [Zevin et al. 2016; Bahaadini et al. 2017] from scratch on our same train-test dataset for a fair comparison, since the Gravity Spy dataset has been recently updated. Note that our Inception and ResNet models are capable of narrowing down any input to within 5 classes with 100.00% accuracy.

We have also provided a comparison between the most popular state-of-the-art models for image classification on the transfer learning problem. Furthermore, we have demonstrated that transfer learning always provides significant improvements over the traditional approach used for classifying spectrograms, and allows the use of the most powerful CNNs on very small datasets, even when they are very dissimilar compared to the original data used for pre-training. Once trained, inference (evaluation on new inputs) can be carried out in a few milliseconds using a GPU, allowing real-time classification of glitches in LIGO.

Note that both the ResNet and Inception models achieved perfect precision and recall on the Paired Doves test set even though it was the smallest class with only 21 elements.
in the training set. Previous methods trained from the ground up had achieved sub-optimal results for this class (Zevin et al., 2016). Therefore, this analysis suggests that the high accuracy is due to transfer learning from the larger dataset, since similar patterns could have been learned when pre-trained on the ImageNet data. Therefore, we have shown that, although the number of samples vary greatly between different classes (up to roughly a factor of 100), we could obtain high precision and recall on even the smallest classes, such as “Paired_Doves”, without employing any class balancing techniques while training.

The best performing models on ImageNet are currently based on ensembles/committees of different CNNs. Since the confusion matrices indicate that each CNN has different strengths and weaknesses for many classes, we expect that using an ensemble of different models would further boost the accuracy for glitch classification. Furthermore, as the Gravity Spy dataset is enlarged in the future with more hand-labeled glitches, our CNNs may be re-trained with the largest available dataset to improve the accuracy.

However, there may be an upper bound set by the error-rate of humans providing the labels. It can be seen in 4.9 that most of the misclassification made by our CNNs were either due to incorrect labeling, or due to superposition of two different classes of glitches, or due to inputs whose true class remain ambiguous. Since our method can be trained very quickly, it can be used to efficiently correct mislabeled elements in the original dataset by training on a large number of randomized train-test splits, and searching for the most commonly mislabeled elements among all these test sets.

The CNNs we have developed in this study may be immediately used in LIGO pipelines for classifying known categories of glitches with high accuracy in real-time. The well-known ability of deep neural networks to generalize (Goodfellow et al., 2016; Lecun et al., 2015) implies that the classifications would be resilient to changes in background noise. Therefore, we expect the excellent performance we achieved on the current data would translate to future observing runs. Nonetheless, the accuracy can be further improved by periodically re-training with larger datasets containing all available labeled examples of glitches at each
time.

After applying transfer learning, i.e., fine-tuning the CNNs on the small dataset of glitches, the CNNs may also be used as good feature extractors for finding new categories of glitches from unlabeled data in an unsupervised or semi-supervised manner. This method, which combines supervised learning, transfer learning, and unsupervised learning, can be used to identify many more categories of noise transients and estimate at what times new types of glitches with similar morphologies start occurring. This may also be used to correct mislabeled glitches in the original dataset used for training/testing by searching for anomalies in the feature-space.

For any of our models, removing the final softmax and fully-connected layer near the output produces a CNN that maps any input image to a vector of real numbers which encode useful information distinguishing different classes of glitches. In this high-dimensional space, glitches having similar morphology will be clustered together. Anomalies and mis-labeled examples will appear isolated from the clusters. This can be visualized by reducing the dimensions of the feature space to 2-D or 3-D using a suitable dimension reduction algorithm (e.g., t-Distributed Stochastic Neighbor Embedding commonly known as t-SNE [van der Maaten & Hinton, 2008]) as shown in 4.4 and 4.5).

Therefore, when new types of glitches appear, which are classified as None_of_the_Above (which means that it does not belong to any of the known classes) by our CNN model, they may be mapped to vectors (representing meaningful features based on the glitch morphology) using these truncated CNN feature extractors. Then, new classes of glitches may be identified by applying standard unsupervised clustering algorithms on these vectors (for example, spectral clustering as shown in Appendix C) and given new automatically generated labels. This automated clustering can also be used to accelerate human labeling of new glitches in a semi-supervised manner by presenting batches of similar glitches to a citizen scientist.

We believe the reason why this clustering technique works very well is because we have used very deep CNNs that are capable of extracting and representing a large number of
highly complex features of the input images. Therefore, transfer learning enables the use of these CNNs which are excellent feature extractors, thus effectively enabling the use of unsupervised learning techniques for finding new classes of glitches. The performance of different clustering techniques on the features generated by these truncated CNNs may be investigated in detail in a subsequent study.

4.5 Conclusion

In this article we have introduced the first application of Deep Learning with Transfer Learning in the context of LIGO detector characterization. Using this method, we have developed state-of-the-art CNNs for LIGO glitch classification using the Gravity Spy dataset. We have shown that by using pre-trained weights of very deep CNNs optimized for real-world object recognition, and then fine-tuning all their layers on a small dataset of labeled glitches, we
Figure 4.5: This is the CNN feature-space projected to 3D using t-SNE. A new simulated class called Reverse_Chirp, which are chirps reflected about the vertical axis, has been added. It can been seen that the CNN feature-extractor maps this class (which it has never seen before during training) to a unique cluster. Furthermore, this cluster is located near the Chirp class and the None_of_the_Above class, which indicates that the relative positions of the glitches in this feature-space is meaningful and depends on their morphology. Note that glitches in the None_of_the_Above are also grouped into smaller clusters, suggesting that they can be easily separated by clustering methods in this feature space.
can significantly increase the accuracy compared to neural networks that have been trained from scratch. Furthermore, the training time is significantly reduced with our approach by several orders of magnitude, and the effort required to design CNN models and optimize their hyper-parameters can be eliminated.

By employing this transfer learning method for glitch classification, we can create a larger dataset of labeled transients with real LIGO data, and the corresponding time-series can be fetched from the GPS times to produce labeled time-series inputs. This dataset may be added to the training process in the Deep Filtering pipeline introduced in (George & Huerta, 2018), to unify detection, classification, and parameter estimation of GW signals directly from time-series data streams for GW detectors in real-time, while being robust in the presence of various types of glitches. Therefore both these techniques can be applied in combination to carry out the full analysis on data collected by GW detectors.

Note that even though the Gravity Spy dataset of spectrograms was not at all similar to the real-world objects found in the ImageNet database, we have demonstrated that the transfer learning approach works extremely well. Therefore, we expect that our method will be useful in general for time-series classification or anomaly detection, for example in analyzing time-series data from telescopes such as Kepler light curves as well as in other fields such as finance, seismology, medicine, etc., after representing the data in the form of images such as spectrograms. Furthermore, we have developed a novel method to encode different duration spectrograms (i.e., multiple views of the same input) as the different RGB color channels in a single image, which renders excellent results and allows the direct use of pre-trained state-of-the-art CNN models. Deep Learning with Transfer Learning using CNNs may also be applied directly for classifying 2D images from telescopes.

The algorithms we have introduced in this article may be used to classify new time-series data in the Gravity Spy project, as well as big datasets from future LIGO observing runs and from international partners, such as Virgo (Acernese et al., 2015), KAGRA (Hirose et al., 2014) and LIGO-India (Unnikrishnan, 2013), as they come online in the next few years. This
would be useful both for improving the data quality in the detectors and for generating vetoes in GW search pipelines. The transfer learning method allows us to use very deep CNNs, with well-known architectures, which are capable of extracting more complex features from the spectrogram images. New classes of glitches may also be found in an unsupervised manner or labeled rapidly in a semi-supervised manner with our method by truncating these CNNs, and using them as feature extractors for clustering algorithms as discussed before. We expect our method for glitch classification and clustering methods may help in finding the instrumental or environmental sources of many classes of glitches whose origins remain unknown, prevent false detection, and enhance the quality of data from the gravitational wave detectors. The fully-trained CNNs developed in this article will be made available to the community for use in glitch classification and detector characterization pipelines.

4.6 Acknowledgements

We thank Gabrielle Allen, Ed Seidel, Scott Coughlin, Vicky Kalogera, Aggelos Katsaggelos, Joshua Smith, Kai Staats, Sara Bahaadini, and Michael Zevin for productive interactions. We thank Kai Staats, Laura Nuttall, the Gravity Spy team, and many others for reviewing this article and providing feedback. We are grateful to NVIDIA for supporting this research by donating four P100 GPUs, to Wolfram Research for offering several Mathematica licenses, and to Vlad Kindratenko for providing dedicated access to a high-end machine at the Innovative Systems Lab at NCSA. We acknowledge the Gravity Spy project and the citizen scientists who participated in it for creating the dataset we used. We also acknowledge the LIGO collaboration for the use of computational resources and for conducting the experiments from which the raw data was obtained.
4.7 Appendix

In this Section we present supplementary information summarizing the networks we experimented with to classify spectrograms of the Gravity Spy dataset. 4.6–4.8 present the confusion matrices for VGG16, VGG19 and InceptionV2 models respectively.

Figure 4.6: Confusion matrix for VGG16. The accuracy is 98.15%.

4.2 presents the recall and precision of our Deep Transfer Learning algorithm for glitch classification. These results clearly exhibit the power of this method, especially for rare
Figure 4.7: Confusion matrix for VGG19. The accuracy is 98.21%.
classes with few labeled examples, therefore the transfer learning method offers a promising framework for future development in glitch classification for GW detectors.

Table 4.2: Table of Precision and Recall on the Test Set

<table>
<thead>
<tr>
<th>Class</th>
<th>Quantity</th>
<th>VGG16 recall</th>
<th>VGG19 recall</th>
<th>ResNet50 recall</th>
<th>InceptionV2 recall</th>
<th>InceptionV3 recall</th>
<th>Previous Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>1080Lines</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>98.51%</td>
<td>98.56%</td>
<td>100.00%</td>
<td>95.65%</td>
<td>100.00%</td>
<td>97.06%</td>
</tr>
<tr>
<td>1400Ripples</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>91.43%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>97.92%</td>
<td>97.91%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>82.69%</td>
</tr>
<tr>
<td>Air_Compressor</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>80.00%</td>
<td>85.71%</td>
<td>85.71%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>85.71%</td>
</tr>
<tr>
<td>Blip</td>
<td></td>
<td>99.47%</td>
<td>98.67%</td>
<td>98.67%</td>
<td>99.99%</td>
<td>99.70%</td>
<td>98.94%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>97.66%</td>
<td>98.94%</td>
<td>99.20%</td>
<td>98.94%</td>
<td>99.73%</td>
<td>98.15%</td>
</tr>
<tr>
<td>Chirp</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>92.86%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>100.00%</td>
<td>100.00%</td>
<td>93.33%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Extremely_Loud</td>
<td></td>
<td>98.91%</td>
<td>98.91%</td>
<td>100.00%</td>
<td>98.91%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Helix</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>98.21%</td>
</tr>
<tr>
<td>Koi_Fish</td>
<td></td>
<td>99.40%</td>
<td>99.40%</td>
<td>98.80%</td>
<td>98.80%</td>
<td>99.40%</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>99.41%</td>
<td>99.40%</td>
<td>98.80%</td>
<td>98.80%</td>
<td>99.40%</td>
<td>99.40%</td>
</tr>
<tr>
<td>Light_Modulation</td>
<td></td>
<td>99.13%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>97.39%</td>
<td>95.65%</td>
<td>99.13%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>97.44%</td>
<td>95.04%</td>
<td>99.14%</td>
<td>99.11%</td>
<td>98.21%</td>
<td>98.28%</td>
</tr>
<tr>
<td>Low_Frequency_Burst</td>
<td></td>
<td>96.97%</td>
<td>96.97%</td>
<td>98.48%</td>
<td>100.00%</td>
<td>99.24%</td>
<td>96.21%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>98.40%</td>
<td>99.20%</td>
<td>98.48%</td>
<td>100.00%</td>
<td>99.01%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Low_Frequency_Lines</td>
<td></td>
<td>97.49%</td>
<td>95.79%</td>
<td>95.79%</td>
<td>100.00%</td>
<td>97.85%</td>
<td>92.55%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>83.78%</td>
<td>86.49%</td>
<td>91.89%</td>
<td>100.00%</td>
<td>97.30%</td>
<td>86.49%</td>
</tr>
<tr>
<td>No_Glitch</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>None_of_the_Above</td>
<td></td>
<td>61.11%</td>
<td>50.00%</td>
<td>83.33%</td>
<td>66.67%</td>
<td>61.11%</td>
<td>44.44%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>78.57%</td>
<td>69.23%</td>
<td>93.75%</td>
<td>92.31%</td>
<td>91.67%</td>
<td>80.00%</td>
</tr>
<tr>
<td>Paired_Doves</td>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Power_Line</td>
<td></td>
<td>97.86%</td>
<td>98.90%</td>
<td>97.80%</td>
<td>98.90%</td>
<td>100.00%</td>
<td>98.90%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>97.86%</td>
<td>98.90%</td>
<td>97.80%</td>
<td>98.90%</td>
<td>100.00%</td>
<td>98.90%</td>
</tr>
<tr>
<td>Repeating_Bips</td>
<td></td>
<td>94.74%</td>
<td>96.49%</td>
<td>98.25%</td>
<td>96.49%</td>
<td>98.25%</td>
<td>89.47%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>100.00%</td>
<td>94.83%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>96.55%</td>
<td>94.41%</td>
</tr>
<tr>
<td>Scattered_Light</td>
<td></td>
<td>95.86%</td>
<td>98.91%</td>
<td>98.91%</td>
<td>98.91%</td>
<td>98.91%</td>
<td>98.91%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>98.91%</td>
<td>100.00%</td>
<td>97.85%</td>
</tr>
<tr>
<td>Scratchy</td>
<td></td>
<td>95.83%</td>
<td>95.83%</td>
<td>95.83%</td>
<td>95.83%</td>
<td>95.83%</td>
<td>97.26%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>95.83%</td>
<td>95.83%</td>
<td>95.83%</td>
<td>95.83%</td>
<td>95.83%</td>
<td>97.26%</td>
</tr>
<tr>
<td>Tomte</td>
<td></td>
<td>98.89%</td>
<td>98.89%</td>
<td>98.89%</td>
<td>98.89%</td>
<td>98.89%</td>
<td>98.89%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>88.89%</td>
<td>90.00%</td>
<td>90.00%</td>
<td>100.00%</td>
<td>88.89%</td>
<td>75.00%</td>
</tr>
<tr>
<td>Violin_Mode</td>
<td></td>
<td>88.89%</td>
<td>90.00%</td>
<td>90.00%</td>
<td>100.00%</td>
<td>88.89%</td>
<td>75.00%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>97.94%</td>
<td>95.88%</td>
</tr>
<tr>
<td>Wandering_Line</td>
<td></td>
<td>88.89%</td>
<td>90.00%</td>
<td>90.00%</td>
<td>100.00%</td>
<td>88.89%</td>
<td>66.67%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>98.90%</td>
<td>98.90%</td>
<td>98.90%</td>
<td>98.90%</td>
<td>98.90%</td>
<td>98.90%</td>
</tr>
<tr>
<td>Whistle</td>
<td></td>
<td>98.33%</td>
<td>98.33%</td>
<td>98.33%</td>
<td>98.33%</td>
<td>98.33%</td>
<td>83.61%</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>98.33%</td>
<td>98.33%</td>
<td>98.33%</td>
<td>98.33%</td>
<td>98.33%</td>
<td>94.44%</td>
</tr>
</tbody>
</table>

The table lists recall and precision for different CNN models. The last column compares the results with the merged-view CNN model described in [Zevin et al. 2016] [Bahadadi et al. 2017] after training and testing it on the exact same train-test split of the updated Gravity Spy dataset, which we used for our models.

4.3 describes in detail the architecture of VGG16, VGG19 and ResNet50, including a snap-
shot of the final structure of these networks. The more complex architecture of the Inception model is shown in 4.10.

Table 4.3: Neural Network Structures

<table>
<thead>
<tr>
<th>VGG16</th>
<th>VGG19</th>
<th>ResNet50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv3-64 × 2</td>
<td>Conv3-64 × 2</td>
<td>(1 × 1, 64) × 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 256</td>
</tr>
<tr>
<td>Conv3-128 × 2</td>
<td>Conv3-128 × 2</td>
<td>(1 × 1, 128) × 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 512</td>
</tr>
<tr>
<td>Conv3-256 × 3</td>
<td>Conv3-256 × 4</td>
<td>(1 × 1, 256) × 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 1024</td>
</tr>
<tr>
<td>Conv3-512 × 6</td>
<td>Conv3-512 × 8</td>
<td>(1 × 1, 512) × 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 512</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 2048</td>
</tr>
<tr>
<td>Max-Pooling</td>
<td>Max-Pooling</td>
<td>Average-Pooling</td>
</tr>
<tr>
<td>FC-4096 × 2</td>
<td>FC-4096 × 2</td>
<td></td>
</tr>
<tr>
<td>FC-22 Softmax</td>
<td>FC-22 Softmax</td>
<td>FC-22 Softmax</td>
</tr>
</tbody>
</table>

VGG16, VGG19 and ResNet50 architectures are shown. The ResNet50 model employs residual (skip) connections between the modules. The final layers of each of the networks were replaced with a new fully connected (FC) layer with 22 neurons followed by the softmax function.

Figure 4.11 presents the architecture of VGG16, and outlines the specific differences with VGG19.

4.12 presents samples of Gravity Spy glitches with the multi-duration encoding used to train and test our Inception models. 4.13 presents a gallery of samples from ImageNet. This database is used in the ILSVRC competitions to benchmark computer vision algorithms, and was used to pre-train the CNN models which we utilized for glitch classification in this article.
Figure 4.8: Confusion matrix for InceptionV2. The accuracy is 98.78%. 

108
Figure 4.9: All misclassified elements in the test set for the InceptionV3 model. The 1s and 4s spectrograms are shown. It can be seen that the CNNs always outputs reasonable classifications with many of the mistakes due to incorrect labels in the test set or glitches whose classes appear ambiguous to even humans. Therefore, the CNN has achieved close to ideal accuracy, i.e, the Bayes error rate. This suggests that mislabeled examples can be found and corrected by trying different randomized test-train splits of the dataset, and then inspecting the mistakes made by the CNN on the test set.
Figure 4.10: Architecture of InceptionV2 and InceptionV3 models. The parallel structure with Inception modules significantly improves evaluation speed and reduces total model size compared to the other CNN models. We used the InceptionV2 model currently in use for the ImageIdentify project pre-trained on a superset of the ImageNet dataset, which is publicly available.
Figure 4.11: Architecture of VGG16. VGG19 has a similar linear design with additional convolution modules as shown in 4.3.
Figure 4.12: Random samples of multi-duration spectrograms encoded in single images. The RGB color channels correspond to 1.0s, 2.0s, and 3.0s durations respectively. This encoding was used to train and test the Inception models. Note that although the color channels appear overlayed to the human eye, the CNN sees this as three independent input images (matrices).
Figure 4.13: Random samples of images from the ImageNet database, which is a common benchmark used to test computer vision algorithms. It contains images of real-world objects belonging to 1000 different classes. Examples of classes include hamster, taxi cab, stingray, tricycle, volleyball, lamp, mushroom, restaurant, water bottle, hook, ladle, kite, speedboat, etc. This dataset was used to pre-train the models used in this article for glitch classification. The pre-trained weights were used to initialize the CNNs before fine-tuning.
Figure 4.14: Unsupervised clustering of glitches belonging to none of the known classes. Note that glitches having similar morphology are located near each other. Our trained CNNs were used as feature extractors to map the images to a high-dimensional vector-space, which in turn was mapped to the 2-D space shown above using the t-SNE dimension reduction method. This feature-extractor can be used to group new classes of glitches by morphology and find the time of their occurrence in the future.
Figure 4.15: Some clusters of glitches in the None_of_the_Above class automatically found by an unsupervised spectral clustering algorithm applied to the CNN feature-space which was shown in the previous Figure. Each row represents a cluster. Different clustering algorithms may be used, and their parameters can be tuned to vary the sensitivity of clustering to enforce stricter similarity criteria.
Chapter 5

Conclusions and Work in Progress

It is evident that GW astronomy, multimessenger astrophysics, HPC simulations, and AI based on deep learning with new hardware architectures, will each play an increasingly important role in the future. This work offers the first opportunity to merge these exciting fields to accelerate scientific discovery by initiating a new paradigm for interdisciplinary research and by integrating expertise in observational astronomy, theoretical and computational physics, data science, signal processing, and computer science to observe and study exotic events that could otherwise go unnoticed. Furthermore, the deep learning methods developed in this thesis for processing highly noisy time-series signals has the potential to be widely applied in other disciplines of engineering, science, and technology, ranging from weather and earthquake prediction to finance.

In the following section we briefly describe some of our ongoing and preliminary work.

5.1 Work in progress

Traditional denoising methods, such as principal component analysis and dictionary learning, are not optimal for dealing with the non-Gaussian, non-stationary noise in the GW detectors especially for low signal-to-noise ratio GW signals. Furthermore, these methods are computationally expensive on large datasets. To overcome these issues, we applied state-of-the-art signal processing techniques, based on recent groundbreaking advancements in deep learning, to denoise gravitational wave signals embedded either in Gaussian noise or in real LIGO noise. We developed SMTDAE, a Staired Multi-Timestep Denoising Autoencoder, based on
sequence-to-sequence bi-directional Long-Short-Term-Memory recurrent neural networks for this task. We demonstrate the advantages of using our unsupervised deep learning approach for denoising and show that, after training only using simulated Gaussian noise, SMTDAE achieves superior recovery performance for gravitational wave signals embedded in real non-Gaussian LIGO noise (Shen et al., 2017).

For multi-messenger astrophysics, it is critical to not only rapidly and accurately detect GW signals, but also to localize their source locations in the sky to trigger observations with astronomical facilities to search for potential EM emission. We have extended our Deep Filtering method to predict the sky localization of GWs using multiple simultaneous inputs from a network of three or more detectors placed in the geographical locations of the current and upcoming GW detectors. Our results show that this method provides state-of-art results, achieving significantly improved sky localization of GW sources compared to previous approaches. We are now extending our deep learning pipeline to build a classifier that can detect and categorize a GW signal into different classes of sources such as binary neutron star mergers and neutron star black hole mergers, supernovae, hypernovae etc., which will can generate extremely useful triggers to astronomers for follow-up campaigns. We have also recently applied the Deep Filtering method to detect GWs produced from supernovae using catalogs of simulations. We achieved the same sensitivity of detection at each SNR as we had obtained with the BBH waveforms. This suggests that our deep learning method can be successfully applied to other types of GW sources given catalogs of templates for training.

Recent work suggests that space-based GW detectors such as the evolved Laser Interferometer Space Antenna (eLISA) (Amaro-Seoane et al., 2012; Gair et al., 2013) will be able to detect stellar mass BBH systems weeks before they merge in the frequency band of ground-based GW detectors (Sesana et al., 2016). DNNs can be used to detect these sources in the eLISA and aLIGO frequency bands using a unified pipeline (on-board analysis may be possible in space with extremely power-efficient chips dedicated for deep learning inference).
Furthermore, by training similar deep learning-based pipelines with DES, LSST, WFIRST, and other open data, we can develop robust, low-latency classification algorithms to search for EM transients in the anticipated sky region where these events are expected to occur.

Another exciting prospect for GW detection is related to sources emitting in the nanoHertz region of the GW spectrum (Hobbs et al., 2010; McLaughlin, 2013), e.g., supermassive black hole binaries (SMBHBs), cosmic strings, and a stochastic GW background generated by a cosmological population of SMBHBs (Arzoumanian et al., 2016; Siemens et al., 2013; Sesana, 2010, 2013; Sesana & Vecchio, 2010; Sesana & Vecchio, 2010). The expected signature of these events is under investigation using analytical (Huerta et al., 2015; Enoki & Nagashima, 2007; Ravi et al., 2014) and statistical frameworks (Taylor et al., 2016b,a). Deep learning may help expand the parameter-space used in these searches, and provide a unified framework to simultaneously target these sources. We are exploring deep learning methods for this task which may enable deep systematic GW searches by also taking into account correlations among pulsars in a detector network.

Deep learning methods can be immediately applied through distributed computing. Therefore, they would aid large-scale projects such as Einstein@Home (Pletsch & Allen, 2009) and SETI@Home (Anderson et al., 2002). Several open-source deep learning libraries, including MXNet, allow scalable distributed training and evaluation of neural networks simultaneously on heterogeneous portable devices thus facilitating citizen science campaigns.

Furthermore, DNNs are particularly suited for image and video processing, therefore, they can be trained to simultaneously search for GW transients and their EM counterparts using raw image data from telescopes. If the identification of an EM transient can be carried out quickly, we can interface this information with a DNN-based GW detection pipeline and vice-versa. Joint analyses of this nature will enable real-time multimessenger astrophysics searches. DNNs can be applied to rapidly identify EM transients in raw image data from state-of-the-art and next-generation astronomical facilities. We are currently experimenting with DES data to train DNNs for this type of analysis. The fast and energy-efficient nature
of embedded GPU inference engines can allow on-site/onboard real-time analysis of big data from survey telescopes, including DES and LSST, and potentially even space-based missions. GW detection, astronomical image processing, and intelligent data reduction may be unified under a single low-latency framework based on deep learning in the future.
References

Aasi, J., Abadie, J., Abbott, B. P., et al. 2015, Classical and Quantum Gravity, 32, 115012
Abbott, B. P., et al. 2016b, Living Reviews in Relativity, 19, 1


Acernese, F., et al. 2015, Classical and Quantum Gravity, 32, 024001


Bengio, Y., & LeCun, Y. 2007, in Large Scale Kernel Machines, ed. L. Bottou, O. Chapelle, D. DeCoste, & J. Weston (MIT Press)


Caruana, R. 1993, in Proceedings of the Tenth International Conference on Machine Learning (Morgan Kaufmann), 41–48

Chen, T., Li, M., Li, Y., et al. 2015, CoRR, abs/1512.01274


Chollet, F., et al. 2015, Web


Chu, T., Fong, H., Kumar, P., et al. 2016, Classical and Quantum Gravity, 33, 165001

Cornish, N. J., & Littenberg, T. B. 2015, Classical and Quantum Gravity, 32, 135012

Dai, W., Dai, C., Qu, S., Li, J., & Das, S. 2016, CoRR, abs/1610.00087


Einstein, A. 1915, Königlich Preussische Akademie der Wissenschaften Zu Berlin, Sitzungberichte, 1915, 844

Enoki, M., & Nagashima, M. 2007, Progress of Theoretical Physics, 117, 241


Fukushima, K. 1980, Biological Cybernetics, 36, 193


George, D., & Huerta, E. 2018, Physical Review D, 97, 044039

Gerosa, D., Kesden, M., Berti, E., O'Shaughnessy, R., & Sperhake, U. 2013, Phys. Rev. D, 87, 104028
Graupe, D. 2013, Principles of Artificial Neural Networks
He, K., Zhang, X., Ren, S., & Sun, J. 2015, CoRR, abs/1512.03385
Hirose, E., Sekiguchi, T., Kumar, R., Takahashi, R., & the KAGRA Collaboration. 2014, Classical and Quantum Gravity, 31, 224004
Hochreiter, S., & Schmidhuber, J. 1997, Neural Computation, 9, 1735
Hornik, K., Stinchcombe, M., & White, H. 1989, Neural Networks, 2, 359

123


Ioffe, S., & Szegedy, C. 2015a, CoRR, abs/1502.03167


Jarrett, K., Kavukcuoglu, K., & Lecun, Y. 2009, What is the Best Multi-Stage Architecture for Object Recognition?


Kingma, D. P., & Ba, J. 2014, CoRR, abs/1412.6980


Kumar, A., & Florêncio, D. 2016, CoRR, abs/1605.02427


LeCun, Y., Bottou, L., Orr, G. B., & Müller, K.-R. 1998c, in Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop (Springer-Verlag), 9–50
Löffler, F., Faber, J., Bentivegna, E., et al. 2012, Classical and Quantum Gravity, 29, 115001
Löffler, F., Faber, J., Bentivegna, E., et al. 2012, Classical and Quantum Gravity, 29, 115001
Mackay, D. J. C. 2003, Information Theory, Inference and Learning Algorithms, 640
McLaughlin, M. A. 2013, Classical and Quantum Gravity, 30, 224008
Minsky, M., & Papert, S. 1969, Perceptrons : an introduction to computational geometry
Moore, C. J., & Gair, J. R. 2014, Physical Review Letters, 113, 251101

Nielsen, M. 2016, Neural Networks and Deep Learning, e-book


Ott, C. D. 2009, Classical and Quantum Gravity, 26, 063001

Owen, B. J., & Sathyaprakash, B. S. 1999, Physical Review D, 60, 022002


Powell, J., et al. 2017, Classical and Quantum Gravity, 34, 034002


Pürrer, M. 2016, Phys. Rev. D , 93, 064041


Rosenblatt, F. 1958, Psychological Review, 65, 386
Ruder, S. 2016, CoRR, abs/1609.04747
Sathyaprakash, B. S., & Schutz, B. F. 2009, Living Reviews in Relativity, 12, 2
Sathyaprakash, B. S., & Schutz, B. F. 2009, Living reviews in relativity, 12, 2
Schmidhuber, J. 2015, Neural Networks, 61, 85
Sesana, A., & Vecchio, A. 2010, Classical and Quantum Gravity, 27, 084016
Sesana, A., & Vecchio, A. 2010, Physical Review D, 81, 104008
Siemens, X., Ellis, J., Jenet, F., & Romano, J. D. 2013, Classical and Quantum Gravity, 30, 224015
Simonyan, K., & Zisserman, A. 2014, CoRR, abs/1409.1556
Smith, M., et al. 2013, Astroparticle Physics, 45, 56


The LIGO Scientific Collaboration, Aasi, J., et al. 2015, Classical and Quantum Gravity, 32, 074001


van der Maaten, L., & Hinton, G. 2008, JMLR, 9, 2579

Veitch, J., et al. 2015a, 91, 042003


Yu, F., & Koltun, V. 2016, in ICLR

Zeng, T., & Ji, S. 2015, in 2015 IEEE International Conference on Data Mining, 579–588


129