

# Deep-Learning Continuous Gravitational Waves

Christoph Dreissigacker,<sup>1,2,\*</sup> Rahul Sharma,<sup>3,1,2</sup> Chris Messenger,<sup>4</sup> and Reinhard Prix<sup>1,2</sup>

<sup>1</sup>*Max Planck Institute for Gravitational Physics (Albert-Einstein-Institute), D-30167 Hannover, Germany*

<sup>2</sup>*Leibniz Universität Hannover, D-30167 Hannover, Germany*

<sup>3</sup>*Birla Institute of Technology and Science, Pilani, India*

<sup>4</sup>*SUPA, School of Physics and Astronomy, University of Glasgow, Glasgow G12 8QQ, United Kingdom*

(Dated: May 6, 2019)

We present a first proof-of-principle study for using deep neural networks (DNNs) as a novel search method for continuous gravitational waves (CWs) from unknown spinning neutron stars. The sensitivity of current wide-parameter-space CW searches is limited by the available computing power, which makes neural networks an interesting alternative to investigate, as they are extremely fast once trained and have recently been shown to rival the sensitivity of matched filtering for black-hole merger signals [1, 2]. We train a convolutional neural network with residual (short-cut) connections and compare its detection power to that of a fully-coherent matched-filtering search using the WEAVE pipeline. As test benchmarks we consider two types of all-sky searches over the frequency range from 20 Hz to 1000 Hz: an “easy” search using  $T = 10^5$  s of data, and a “harder” search using  $T = 10^6$  s. Detection probability  $p_{\text{det}}$  is measured on a signal population for which matched filtering achieves  $p_{\text{det}} = 90\%$  in Gaussian noise. In the easiest test case ( $T = 10^5$  s at 20 Hz) the DNN achieves  $p_{\text{det}} \sim 88\%$ , corresponding to a loss in sensitivity depth of  $\sim 5\%$  versus coherent matched filtering. However, at higher-frequencies and longer observation time the DNN detection power decreases, until  $p_{\text{det}} \sim 13\%$  and a loss of  $\sim 66\%$  in sensitivity depth in the hardest case ( $T = 10^6$  s at 1000 Hz). We study the DNN generalization ability by testing on signals of different frequencies, spindowns and signal strengths than they were trained on. We observe excellent generalization: only five networks, each trained at a different frequency, would be able to cover the whole frequency range of the search.

## I. INTRODUCTION

Gravitational waves from binary mergers are now being observed routinely [3–6] by the Advanced LIGO [7] and Virgo [8] detectors. In contrast, the much weaker and longer-lasting (days–months) narrow-band *continuous gravitational waves* (CWs) from spinning deformed neutron stars are yet to be detected, despite a multitude of searches over the past decade (see [9–11] for reviews) and continuing improvements in search methods (e.g. see [12] for a recent overview).

A key limitation of current search methods for CWs with unknown parameters is the “exploding computing cost problem”: give that a putative signal would be very weak, one needs to integrate as much data as possible in order to increase the signal-to-noise ratio (SNR). However, for a fully-coherent matched-filtering search (which is close to statistically optimal [13]), the corresponding computing cost grows as a high power  $\sim T^n$  of the data timespan  $T$ , with  $n \gtrsim 5$ . This typically limits the longest coherent duration to days–weeks before the computing cost would become infeasible.

Therefore the class of semi-coherent methods has been developed, producing computationally cheaper searches. They allow the analysis of more data, typically resulting in better sensitivity than a corresponding coherent search at fixed computing cost (e.g. see [14, 15]). Such methods combined with massive amounts of computing

power, either via local compute clusters or via the distributed public computing platform Einstein@Home [16], currently yield the best state-of-the-art sensitivity to CW signals (e.g. see [17–19] for recent examples).

In this work we investigate *deep neural networks* (DNNs) [20–22] as a novel search method for CWs. The field of DNNs, also referred to as *Deep Learning*, has emerged as one of the most successful machine-learning paradigms in the last decade, dominating wide-ranging fields [22] such as image recognition, speech recognition and language translation, as well as certain board [23] and video games [24, 25].

More recently DNNs have started to draw attention in the field of gravitational-wave searches: (i) as a classifier for non-Gaussian detector transients (*glitches*) [26–29], (ii) as a search method for unmodelled burst signals [30, 31] in time-frequency images produced by coherent WaveBurst [32], and (iii) as a direct detection method for black-hole merger signals in gravitational-wave strain data [1, 2, 33–36].

This last approach (iii) is of particular interest to us, as [1, 2] have illustrated for the first time that DNNs can achieve a detection power comparable to that of (near-optimal) matched filtering, at a fraction of the search time. This is relevant for CW searches: while semi-coherent methods for wide-parameter-space searches are the most sensitive approach currently known, they are by design less sensitive than the statistical optimum achievable according to the Neyman-Pearson-Searle lemma [37].

With DNNs the computationally expensive step is shifted to the *preparation* stage of the search: the archi-

\* christoph.dreissigacker@aei.mpg.de

texture tuning and “learning” of optimal network weights (i.e. the *training*), while the execution time on given input vectors is very short (typically fractions of a second). Determination of the noise-distribution (for estimation of the false alarm level  $p_{\text{fa}}$ ) and measurement of upper limits require many repeated searches over different input data sets, with and without injected signals. The relative search speed advantage of DNNs compared to traditional search methods therefore accumulates dramatically over these operations allowing very fast and flexible search characterizations.

The plan of this paper is as follows: In section II we define and characterize our test benchmarks. In section III we describe our deep-learning approach to searching for continuous gravitational waves: explaining the network architecture and how it was trained. In section IV we characterize the performance our DNN achieves on the test benchmarks in comparison to the matched-filtering performance and how it generalizes beyond the benchmarks’ search parameters. And finally we discuss these results in section V.

## II. COMPARISON TEST BENCHMARKS

### A. Benchmark definitions

In order to characterize the detection power of the DNN that we introduce in the next section, we define two benchmark search setups and measure the corresponding sensitivity achieved on them with a classical (near-optimal) matched-filter search method described in Sec. II B.

We compare the sensitivity in the Neyman-Pearson sense, also known as the receiver-operator characteristic (ROC), using the “upper limit” conventions used in most CW searches (cf. [12]): measure the detection probability  $p_{\text{det}}$  at a chosen false-alarm level  $p_{\text{fa}}$  for a signal population of fixed amplitude  $h_0$ , with all other signal parameters (i.e. polarization, sky-position, frequency and spindown) drawn randomly from their priors. In order to characterize the signal strength in noise, we use *sensitivity depth*  $\mathcal{D}$  [12, 38], defined as

$$\mathcal{D} \equiv \frac{\sqrt{S_n}}{h_0}, \quad (1)$$

where  $S_n$  is the power-spectral density (PSD) of the (Gaussian) noise at the signal frequency, and  $h_0$  is the signal amplitude. In particular we are interested in the sensitivity depth  $\mathcal{D}^{90\%}$  that corresponds to the signal amplitude  $h_0^{90\%}$  at which the search yields a detection probability of  $p_{\text{det}} = 90\%$  at a fixed false-alarm level, which here is chosen as  $p_{\text{fa}} = 1\%$  per 50 mHz frequency band.

We consider two *all-sky* searches (parameters summarized in Table. I) over a range in frequency  $f$  and first-order spindown  $\dot{f}$ , one using  $T = 10^5$  s  $\sim 1.2$  days, and one using  $T = 10^6$  s  $\sim 12$  days of data assuming a single

data span	$T = 10^5$ s / $T = 10^6$ s
detectors	LIGO Hanford
noise	stationary, white, Gaussian
sky-region	all-sky
frequency band	$f \in [20, 1000]$ Hz
spin-down range	$\dot{f} \in [-10^{-10}, 0]$ Hz/s

TABLE I. Definition of the two benchmark searches.

detector (chosen as LIGO Hanford). These two searches could realistically be performed with coherent matched filtering. The required computing cost for the search and its characterization (upper limits, false alarm level) however would still require a large cluster of, say,  $\mathcal{O}(1000)$  cores for over a month or so (see Table. II). Therefore actually performing these two full searches only for the purpose of characterizing the matched-filtering sensitivity would be infeasible. Instead we characterize the matched-filter search on only five narrow frequency bands of width  $\Delta f = 50$  mHz starting at frequencies  $f_0 = 20, 100, 200, 500$  and 1000 Hz, yielding a total of ten representative test cases.

### B. Weave matched-filtering sensitivity

For the matched-filter search we use the recently-developed WEAVE code [39], which implements a state-of-the-art CW search algorithm [40] based on summing coherent  $\mathcal{F}$ -statistics [41, 42] over semi-coherent segments on optimal lattice-based template banks [43, 44]. This code can also perform fully-coherent (i.e. single-segment)  $\mathcal{F}$ -statistic searches, which we use for the present proof-of-principle study. The benchmark search definitions in Table I are chosen in such a way that a fully-coherent search is still computationally feasible. This yields a simpler and cleaner comparison than using a semi-coherent search setup, as we can easily design near-optimal search setups (by using relatively fine template banks) without the extra complication of requiring costly sensitivity-optimization at fixed computing cost [15, 40, 45].

The WEAVE template banks are characterized by a maximal-mismatch parameter  $m$ , which controls how fine the templates are spaced in parameter space. These are chosen as  $m = 0.1$  and  $m = 0.2$  for the two searches with  $T = 10^5$  s and  $T = 10^6$  s, respectively. The reason for choosing the larger mismatch value (i.e. coarser template bank) in the  $T = 10^6$  s case is to keep the computing cost of the corresponding test-cases still practically manageable, as the coherent cost scales with mismatch parameter as  $\propto m^2$  for a four-dimensional template bank (e.g. see Eq.(24) in [43]).

By repeated injections of signals in the data and recovery of the loudest  $\mathcal{F}$ -statistic candidate in the template bank, one can measure the relative SNR-loss  $\mu$  compared to a perfectly-matched template. The resulting measured average mismatch  $\langle \mu \rangle$  quantifies in some sense how close

to “optimal” the matched-filter sensitivity will be (compared to an infinite-computing cost search with  $m = 0$ ), and is found as  $\langle\mu\rangle \sim 5\%$  and  $\langle\mu\rangle \sim 11\%$ , respectively for the two searches.

Using the template-counting and timing models [39, 46, 47] for WEAVE and the resampling  $\mathcal{F}$ -statistic, we can estimate the total number of templates and the corresponding total runtime for these two benchmark searches as  $\sim 78$  days and  $\sim 45\,000$  days on a single CPU core, respectively. Table. II provides a summary of the WEAVE search parameters and characteristics.

name	$T = 10^5$ s	$T = 10^6$ s
mismatch parameter $m$	0.1	0.2
average SNR loss $\langle\mu\rangle$	5%	11%
Number of templates $\mathcal{N}$	$4 \times 10^{11}$	$3 \times 10^{14}$
Search time [single CPU core]	$6.7 \times 10^6$ s	$3.9 \times 10^9$ s

TABLE II. WEAVE parameters and characteristics for the two searches.

In order to estimate the sensitivity for the ten test cases defined in the previous section (i.e. five frequency slices of  $\Delta f = 50$  mHz for each search of  $T = 10^5$  s and  $T = 10^6$  s, respectively), we first determine the corresponding detection thresholds  $\mathcal{F}_{\text{th}}$  on the  $\mathcal{F}$ -statistic corresponding to a false-alarm level of  $p_{\text{fa}} = 1\%$  for each case. This is done by repeatedly ( $10^5$  times for  $T = 10^5$  s, and  $\sim 10^4$  times for  $T = 10^6$  s, respectively) performing each search over Gaussian noise and thereby recording the distribution of the loudest candidate, which yields the relationship between threshold and false-alarm level. The corresponding detection probability  $p_{\text{det}}$  for any given signal population of fixed  $\mathcal{D}$  is obtained by injecting signals into Gaussian noise data and measuring how many times the loudest candidate exceeds the detection threshold. By varying the injected  $\mathcal{D}$  we can eventually find  $\mathcal{D}^{90\%}$  for the desired  $p_{\text{det}} = 90\%$  (e.g. see [12] for more details and discussion of this standard “upper limit” procedure). By a final injection+recovery Monte-Carlo we can verify that the achieved WEAVE detection probability for the quoted thresholds and signal strengths  $\mathcal{D}^{90\%}$  in Table. III is  $p_{\text{det}} \sim 90\% - 91\%$ , which is sufficiently accurate for our present purposes.

The sky template resolution grows as  $\propto f^2$  as a function of frequency  $f$ , resulting in a corresponding increase in the number of templates at higher frequency. This increases the number of “trials” in noise at the higher-frequency slices, which results in a corresponding increased false-alarm threshold (chosen in order to keep the false-alarm level at  $p_{\text{fa}} = 1\%$ ) as well as an increased computing cost, shown in Table. III.

$f_0$	20 Hz	100 Hz	200 Hz	500 Hz	1000 Hz
$T = 10^5$ s					
$\mathcal{N}_{\Delta f}$	$5 \times 10^5$	$1 \times 10^7$	$5 \times 10^7$	$3 \times 10^8$	$1 \times 10^9$
CPU $_{\Delta f}$ [s]	0.1	4.9	19	$2.3 \times 10^2$	$1.7 \times 10^3$
$\mathcal{F}_{\text{th}}(p_{\text{fa}})$	20.6	23.6	25.1	27.0	28.6
$T = 10^6$ s					
$\mathcal{N}_{\Delta f}$	$3 \times 10^8$	$8 \times 10^9$	$3 \times 10^{10}$	$2 \times 10^{11}$	$8 \times 10^{11}$
CPU $_{\Delta f}$ [s]	45	$3 \times 10^3$	$1.4 \times 10^4$	$1.6 \times 10^5$	$6.9 \times 10^5$
$\mathcal{F}_{\text{th}}(p_{\text{fa}})$	27.5	31.1	32.5	34.2	36.2

TABLE III. WEAVE characteristics for the ten test cases, each covering a frequency “slice” of  $\Delta f = 50$  mHz, starting at  $f_0$ , of the full searches defined in Table. I. The detection thresholds  $\mathcal{F}_{\text{th}}$  correspond to a false-alarm level of  $p_{\text{fa}} = 1\%$  over the band  $\Delta f$ . CPU $_{\Delta f}$  denotes the search time in seconds for the respective  $\Delta f$  band on a single CPU core.

$\mathcal{D}^{90\%} [\text{Hz}^{-1/2}]$	$f_0 = 20$ Hz	100 Hz	200 Hz	500 Hz	1000 Hz
$T = 10^5$ s	11.4	10.8	10.4	9.9	9.7
$T = 10^6$ s	29.3	28.2	27.6	26.8	26.0

TABLE IV. Measured WEAVE “upper limit” sensitivity  $\mathcal{D}^{90\%}$  at false-alarm level of  $p_{\text{fa}} = 1\%$ .

### III. DEEP-LEARNING CWS

Our general approach is similar to that of [1, 2] in that we directly use the detector strain data as our network input, and train a simple classifier with two output neurons for the classes “noise” and “signal (in noise)”. However, given that CW signals are long in duration and narrow in frequency, instead of using the time-series input it makes more sense in our case to use the frequency-domain representation of that data. We therefore provide the real- and imaginary parts of the fast Fourier transform (FFT) of the data as a two-dimensional input vector over frequency bins, using the native FFT resolution of  $1/T$ . We chose the network input size to be sufficiently large to contain the widest signal (signals get stretched in frequency domain by spindown  $\dot{f}$  and Doppler shifts) *twice*, so that we can slide the network along the frequency axis in steps of half the network input width, guaranteeing that one input window will always contain the full signal.

#### A. Network architecture

We started experimenting with DNN architectures similar to those described in [1, 2], but eventually by trial and error converged on a ResNet architecture [48], which showed better performance for our problem cases.

We have chosen slightly different networks for the two

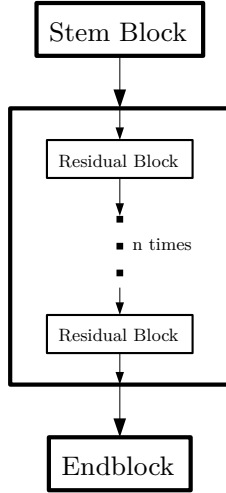


FIG. 1. Illustration of the general network architecture used in this study.

searches ( $T = 10^5$  s and  $T = 10^6$  s) of Table. I, as these correspond to signals with rather different width in frequency domain: the network in the  $T = 10^5$  s cases contains six instances of a residual block, while in the  $T = 10^6$  s cases the network uses twelve.

The network layers can be separated into three parts: the stem block, a block of multiple residual blocks, and an end-block, see Fig 1. The stem block consists of a standard convolutional layer, while each of the residual blocks is built according to [48]. The endblock contains a dense softmax layer with two final output neurons, corresponding to the estimated probability  $p_{\text{signal}}$  that the input contains a signal, and  $p_{\text{noise}} = 1 - p_{\text{signal}}$  for pure noise sample. The DNNs are implemented in the Keras framework[49] on top of a Tensorflow[50] backend.

### B. DNN training and validation

Training the network is performed on a synthesized data-set of input vectors, where half contain pure Gaussian noise, and half contain a signal added to the noise. One full pass through this training set is commonly referred to as a training *epoch*. Using a pre-computed set of 10 000 signals, each signal is added to 24 dynamically-generated noise realizations, which are also used as pure-noise inputs. The number of signals in the training set was determined empirically, as using more signals gives diminishing performance improvements (see figure 2). The signals are scaled to a fixed depth  $\mathcal{D}_{\text{training}}^{90\%}$  for each test case and randomly shifted in frequency within the network input window. These training depths were estimated semi-analytically using the method of [12, 47], and differ slightly from the final measured values  $\mathcal{D}^{90\%}$  of Table. IV, which had not yet been available at the time of training. When testing the network on signals of

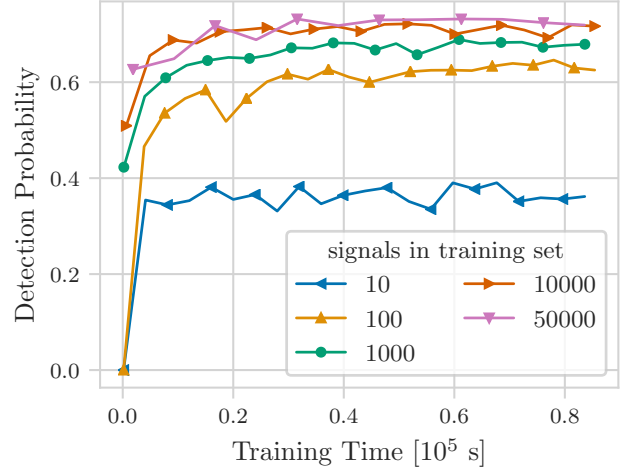


FIG. 2. Validation detection probability for  $T = 10^5$  s,  $f_0 = 1000$  Hz for training with training sets containing 10, 100, 1000, 10000 and 50000 signals.

different depths, the detection probability behaves as expected, see Sec. IV D. Furthermore, we found that using a different choice of training depth did not significantly affect training success.

Every few epochs of training, we perform a *validation* step, where the detection probability of the network is measured on an independent data set. This validation set contains another 20 000 input vectors, half containing signals in noise (of fixed depth  $\mathcal{D}^{90\%}$ ), and half containing noise only.

In order to compute the network’s detection probability  $p_{\text{det}}^{\text{DNN}}$ , we treat the output neuron  $p_{\text{signal}}$  as a statistic, and follow the usual “upper limit” procedure described in Sec. II B: we repeatedly run the network on Gaussian noise inputs in order to determine the  $p_{\text{fa}} = 1\%$  detection threshold. We then run the network on the signal set and measure for what fraction of signals the statistic exceeds that threshold.

The evolution of the detection probability as a function of training epoch (or similarly, as a function of training time) is presented in Fig. 3, illustrating the progress of learning. In order to test the variability and dependence of the learning success on the random initialization of the network, we train a “cloud” of  $\sim 100$  differently-initialized network instances. We use the network at its point of best validation performance from each test case for the further test results presented in the next sections.

Most of the training was performed on Nvidia GTX 750 GPUs. We see in Fig. 3 that for most cases the improvements in detection probability seem to have leveled off after the training time (about one day in the  $T = 10^5$  s cases, and about 10 days in the  $T = 10^6$  s cases). However, in the case of  $T = 10^6$  s,  $f_0 = 1000$  Hz seen in Fig.3d (and also for  $T = 10^6$  s,  $f_0 = 500$  Hz, not shown), there still seems to be a slowly increasing trend in detection probability at the end of training time. Therefore we

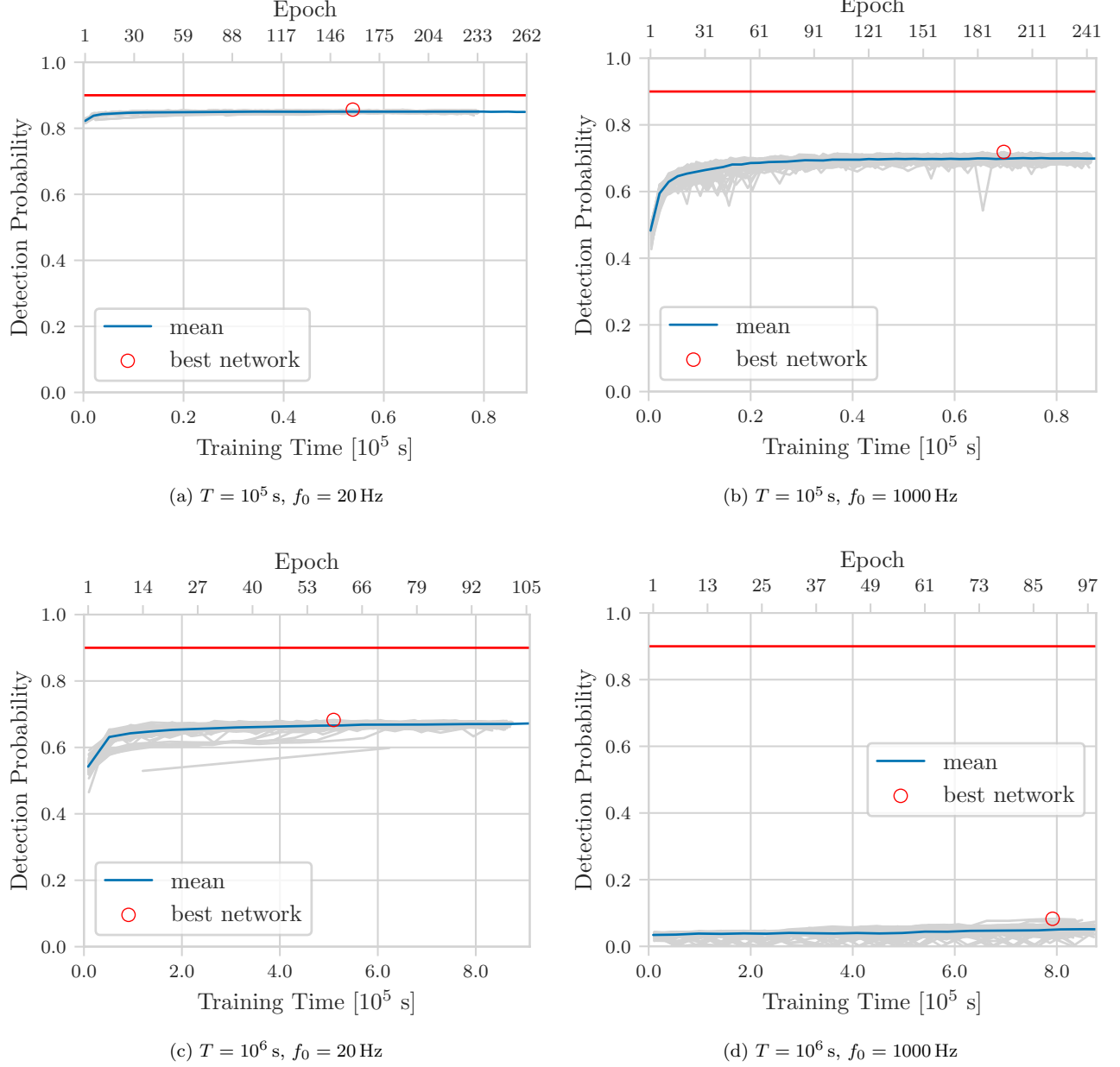


FIG. 3. Validation detection probability  $p_{\text{det}}^{\text{DNN}}$  of the DNN versus training time (or mean trained epoch) for 100 different network instances trained for each of four test cases: **(a)**  $T = 10^5$  s,  $f_0 = 20$  Hz, **(b)**  $T = 10^5$  s,  $f_0 = 1000$  Hz, **(c)**  $T = 10^6$  s,  $f_0 = 20$  Hz and **(d)**  $T = 10^6$  s,  $f_0 = 1000$  Hz, all trained on Nvidia GTX 750. The solid horizontal line denotes the matched-filtering detection performance of  $p_{\text{det}} = 90\%$ .

trained a single network instance for these two cases again on a more powerful Nvidia TITAN V GPU for many more epochs, until the validation detection probability seemed to level off, which is shown in Fig. 4.

Overall we observe an dramatic increase in “difficulty” the DNN has in learning the different test cases along the direction of increasing data span  $T$  and frequency  $f$ , also seen clearly in Table. V. In the easiest case of  $T = 10^5$  s,  $f_0 = 20$  Hz the DNN achieves a detection probability of  $p_{\text{det}}^{\text{DNN}} \sim 88\%$ , nearly rivalling matched-

filtering performance, while in the hardest case of  $T = 10^6$  s,  $f_0 = 1000$  Hz it only manages  $p_{\text{det}} \sim 13\%$  (also see Table. V). This may not be very surprising, given that the cases become increasingly more compute-intensive (more templates) along the same axis for matched filtering, as seen in Table. III. In the frequency-domain input vectors of the DNN, this would manifest by the signals being more widely spread-out due to increased frequency drift  $\dot{f}T$  and Doppler stretching.

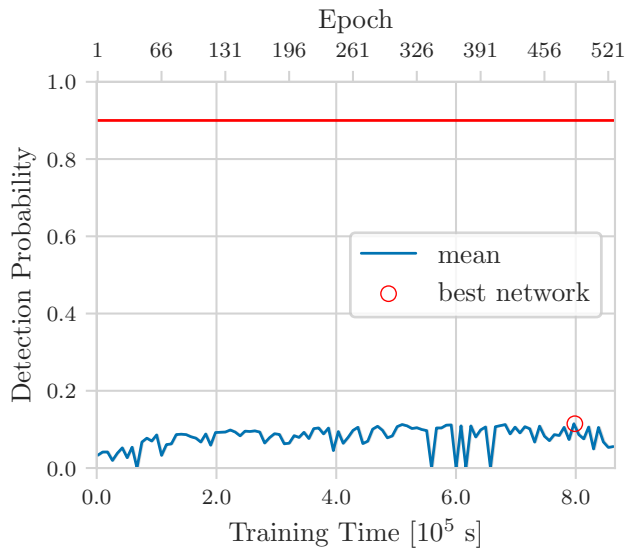


FIG. 4. Validation detection probability  $p_{\text{det}}^{\text{DNN}}$  of the DNN versus training time for a single network trained on Nvidia TITAN V for the case  $T = 10^6$  s,  $f_0 = 1000$  Hz.

#### IV. TESTING DNN PERFORMANCE

After the training and validation steps, we perform a series of tests on the best DNN found for each test case (i.e. with the highest  $p_{\text{det}}^{\text{DNN}}$  over all validation steps), in order to more fully characterize its performance as a CW detection method. In these tests we simulate the signals and noise directly for any given depth using the standard CWLALSUITE[51] machinery, in order to independently verify the network performance. Hence we are not using a traditionally fixed *testing set* but generate it on demand.

##### A. Verifying detection probabilities

As a sanity check we measure again the detection probability  $p_{\text{det}}^{\text{DNN}}$  at  $p_{\text{fa}} = 1\%$  for the ten cases over the respective frequency bands for a signal population at the matched-filtering  $\mathcal{D}^{90\%}$  of Table. IV. The resulting DNN test results obtained with the independent test-pipeline are given in Table. V. These results usually agree to  $\sim 2$

$p_{\text{det}}^{90\%}$	$f_0 = 20$ Hz	100 Hz	200 Hz	500 Hz	1000 Hz
$T = 10^5$ s	$87.6^{+0.7}_{-0.6}$	$85.4^{+0.7}_{-0.7}$	$84.1^{+0.7}_{-0.7}$	$80.2^{+0.8}_{-0.8}$	$73.0^{+0.9}_{-0.9}$
$T = 10^6$ s	$68.8^{+0.9}_{-0.9}$	$50.0^{+1.0}_{-1.0}$	$38.7^{+0.9}_{-1.0}$	$25.4^{+0.8}_{-0.9}$	$13.1^{+0.6}_{-0.7}$

TABLE V. Detection probabilities in % of the best networks for each case at false alarm level  $p_{\text{fa}} = 1\%$  and 90 % matched-filtering depth.

percentage points in detection probability with the corre-

sponding best  $p_{\text{det}}^{\text{DNN}}$  originally observed in the validation step, seen in Figs. 3,4.

A second interesting question is how the detection probability depends on the false-alarm level  $p_{\text{fa}}$  (commonly referred to as ROC curve) for a fixed signal population. This is shown in Fig. 5 in comparison to the matched-filter ROC.

##### B. Generalization in frequency $f$

If we want to perform a search over the whole frequency range (e.g. as defined in Table. I) using DNNs, we would need to determine how many different networks we have to train in order to cover this range with a reasonable overall sensitivity. Alternatively we can also train a single DNN with signals drawn from the full frequency range of the search and compare its performance.

The results of these tests are shown in Fig. 6, which show how the five DNNs, trained at their respective frequencies  $f_0$ , perform over the full frequency range of the search. In addition we show the performance of another network that has been trained directly over the full frequency range.

We see that the “specific” networks trained only on a narrow frequency range still perform reasonably well over a fairly broad range of frequencies, and especially that networks trained at higher frequencies generalize well to lower frequencies. This result shows that a small number of networks  $\mathcal{O}(5)$  would be able to cover the whole frequency range at a similar detection performance that was obtained on the individual training frequencies. Furthermore, for the  $T = 10^5$  s search, it seems quite feasible to train a single network over the full frequency range directly, achieving similar (albeit lower) performance to the “specialized” networks trained at narrow frequency bands. On the contrary for the  $T = 10^6$  s search the detection probability of the “full-range” network drops up to 20 percentage points against the “specialized” networks.

##### C. Generalization in spin-down $\dot{f}$

A further interesting aspect to consider is how far in spindown  $\dot{f}$  the performance network extends beyond the range that it was trained on, i.e.  $\dot{f} \in [-10^{-10}, 0]$  Hz/s as given in Table. I. This is shown in Fig. 7. We see that the DNN detection probability remains high even for spindowns that are 1-2 orders of magnitude larger than the training range. In particular, networks trained at higher frequencies seem to have a wider generalization range in spindown, which makes sense as they would have learned to recognize signal shapes with larger Doppler broadening, a qualitatively similar effect to having more spindown.

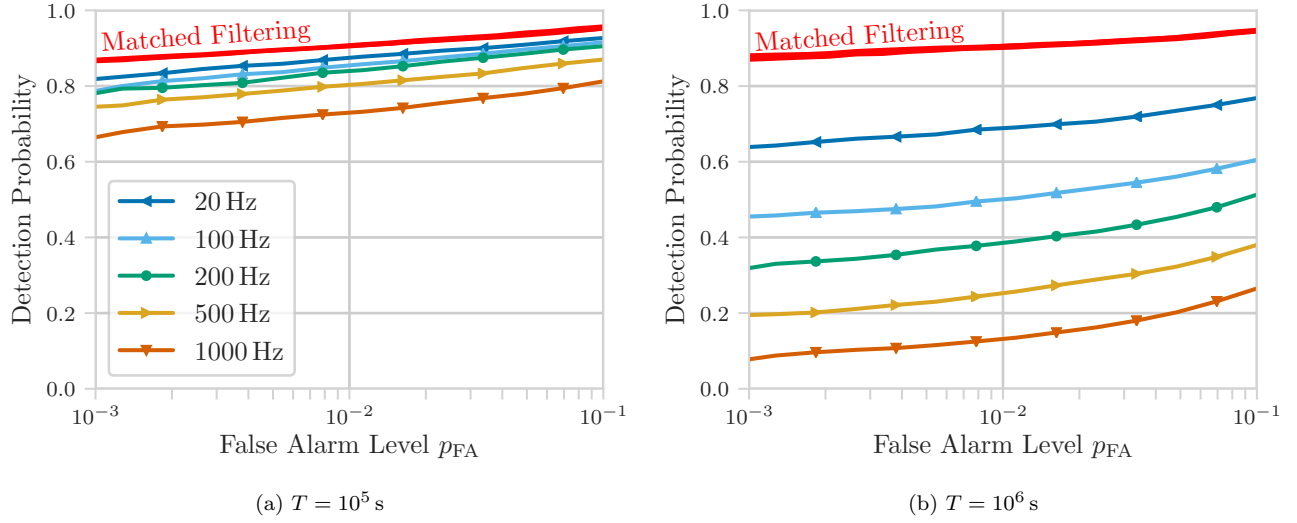


FIG. 5. ROC-curve: Detection probability  $p_{det}$  versus  $p_{fa}$  for the  $10^5$  s search (left) and the  $10^6$  s search (right). The solid red lines indicate the measured ROC curves for matched filtering.

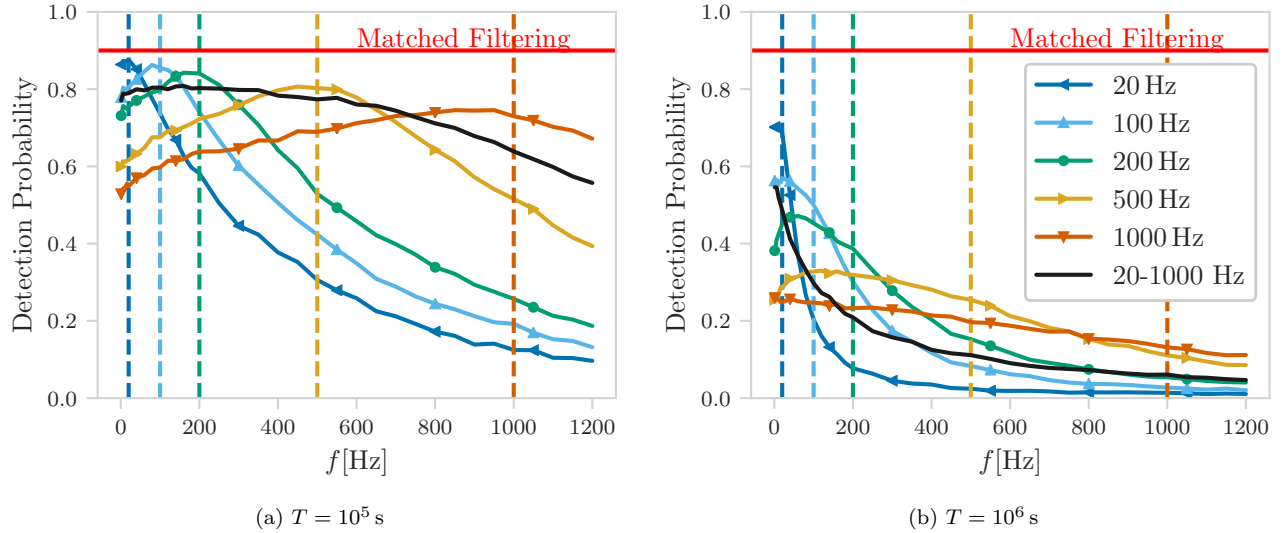


FIG. 6. Detection probability  $p_{det}$  versus injection frequency  $f$  for networks trained at five different frequencies and for a network trained with signals drawn from the full frequency range (solid black line). The dashed vertical lines mark the respective training frequencies for the five “specialized” networks. The horizontal dashed line represents the coherent matched filtering detection performance.

#### D. Generalization in signal strength

Another important issue is how well the DNN generalizes for signals of different strength  $\mathcal{D}$ , given that we only trained each network at one specific depth  $\mathcal{D}_{training}^{90\%}$ , an estimate of the matched filtering depth. The results of this test are shown in the efficiency plots of Fig. 8. We see that generally the dependence of  $p_{det}(\mathcal{D})$  for the DNNs seems to be quite similar to that of matched filtering, but shifted to its overall (lower) performance level.

Conversely we also calculated the “upper limit” sensitivity depth  $\mathcal{D}_{DNN}^{90\%}$  where the network achieves 90% detection probability (see Table VI). These values correspond to a sensitivity loss of 5% – 21% (as a function of frequency) for the  $T = 10^5$  s search, and 26% – 66% for the  $T = 10^6$  s search.



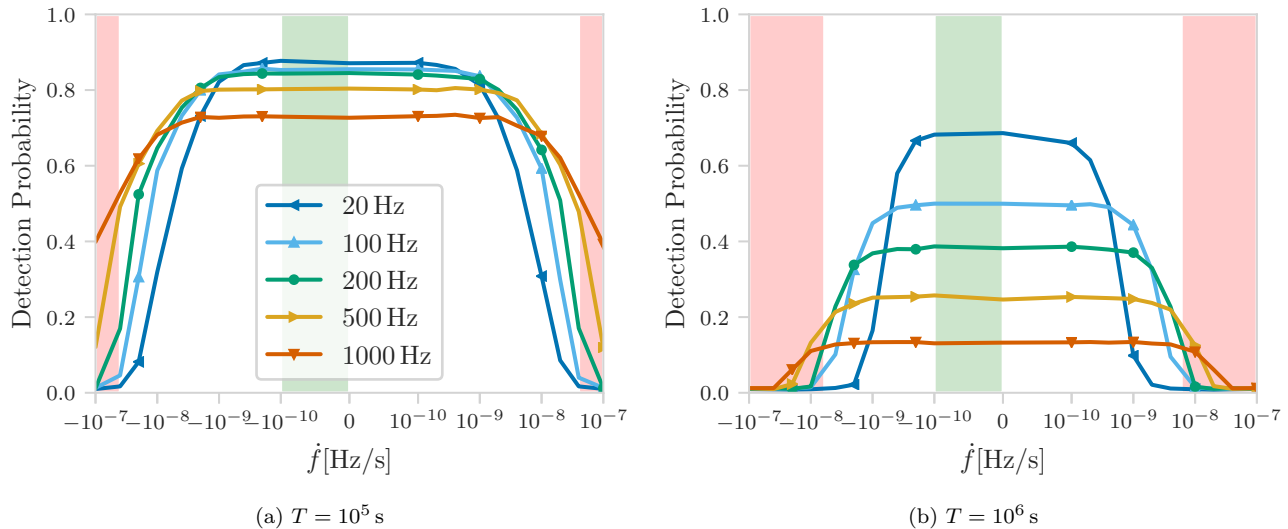


FIG. 7. Detection probability  $p_{\text{det}}$  versus injected spin-down  $\dot{f}$  for networks trained at give different frequencies. The green shade in the middle marks the  $10^{-10}$  Hz/s wide spin-down band the networks were trained on. The x-axis is plotted as a symmetric logarithm, i.e. logarithmical for the larger negative values, linear for  $|\dot{f}| < 10^{-10}$  Hz/s and logarithmical for the larger positive values. The red shades at the edges illustrates where we start losing SNR purely by the network input window being smaller than the widest signals.

$\mathcal{D}_{\text{DNN}}^{90\%} [\text{Hz}^{-1/2}]$	$f_0 = 20 \text{ Hz}$	100 Hz	200 Hz	500 Hz	1000 Hz
$T = 10^5 \text{ s}$	10.8	10.0	9.5	8.6	7.7
$T = 10^6 \text{ s}$	21.6	16.5	14.3	11.1	8.9

TABLE VI. Sensitivity depths  $\mathcal{D}_{\text{DNN}}^{90\%}$  at false-alarm level of  $p_{\text{fa}} = 1\%$  achieved by the network for the ten test cases. The respective matched filter depths can be found in Table IV.

### E. Timing

The total amount of computational resources needed, is another interesting point of comparison to a matched filter search. The total search times for using the matched-filter WEAVE method on the two benchmark searches can be found in Table. II.

For the DNN the total computation time consists of two parts: Training time and *prediction* time (i.e. calculating one output statistic  $p_{\text{signal}}$  for one input data vector). The training time for the two network architectures is  $\sim 1 \text{ d}$  and  $\sim 10 \text{ d}$  per network for the  $T = 10^5 \text{ s}$  and  $T = 10^6 \text{ s}$  cases, respectively. Only part of this time is actually spent on training the network, another part is calculating the detection probability of the network every few epochs in order to monitor the progress of training.

The prediction time in comparison is almost negligible. The smaller networks for the  $T = 10^5 \text{ s}$  cases require  $\sim 3 \text{ ms}$  for processing one input window. The larger networks for the  $T = 10^6 \text{ s}$  cases need  $\sim 10 \text{ ms}$  per prediction. Each search requires a different number of sliding

input windows to cover the whole frequency range, and the total search time can be found in table VII.

An important detail to note in a direct comparison between matched filtering and a pure classifier “signal” vs “noise” DNN search is that matched filtering yields far more information on outlier candidates that cross the threshold. In particular, its signal parameters will be well constrained already, allowing a follow-up search to be performed in a small region of parameter space. The DNN classifier, on the other hand, would flag input windows (of width  $\Delta f_{\text{IW}}$ ) in frequency as outliers to be followed up. Assuming we follow up two input windows per candidate, one can estimate the total expected follow-up cost (using matched-filtering) as a fraction  $2(\Delta f_{\text{IW}}/\Delta f) p_{\text{fa}}$  of the total matched-filtering cost (see Table II), where  $p_{\text{fa}} = 1\%$  is the false-alarm probability per  $\Delta f = 50 \text{ mHz}$  band.

Cost [s]	Training	Search	Follow-up	Total
$T = 10^5 \text{ s}$	$4.3 \times 10^5$	58.8	$2.2 \times 10^4$	$4.5 \times 10^5$
$T = 10^6 \text{ s}$	$4.3 \times 10^6$	196	$6.5 \times 10^7$	$6.9 \times 10^7$

TABLE VII. DNN computing cost (in seconds) for training, search and follow-up (using matched-filtering). The respective matched-filtering cost can be found in Table II

Therefore even including all the training time and assuming a matched-filter follow-up, the DNN search would still seem to be requiring less computing power. At the present stage, however, we cannot realise this potential benefit given that our DNN search so far is far less sen-



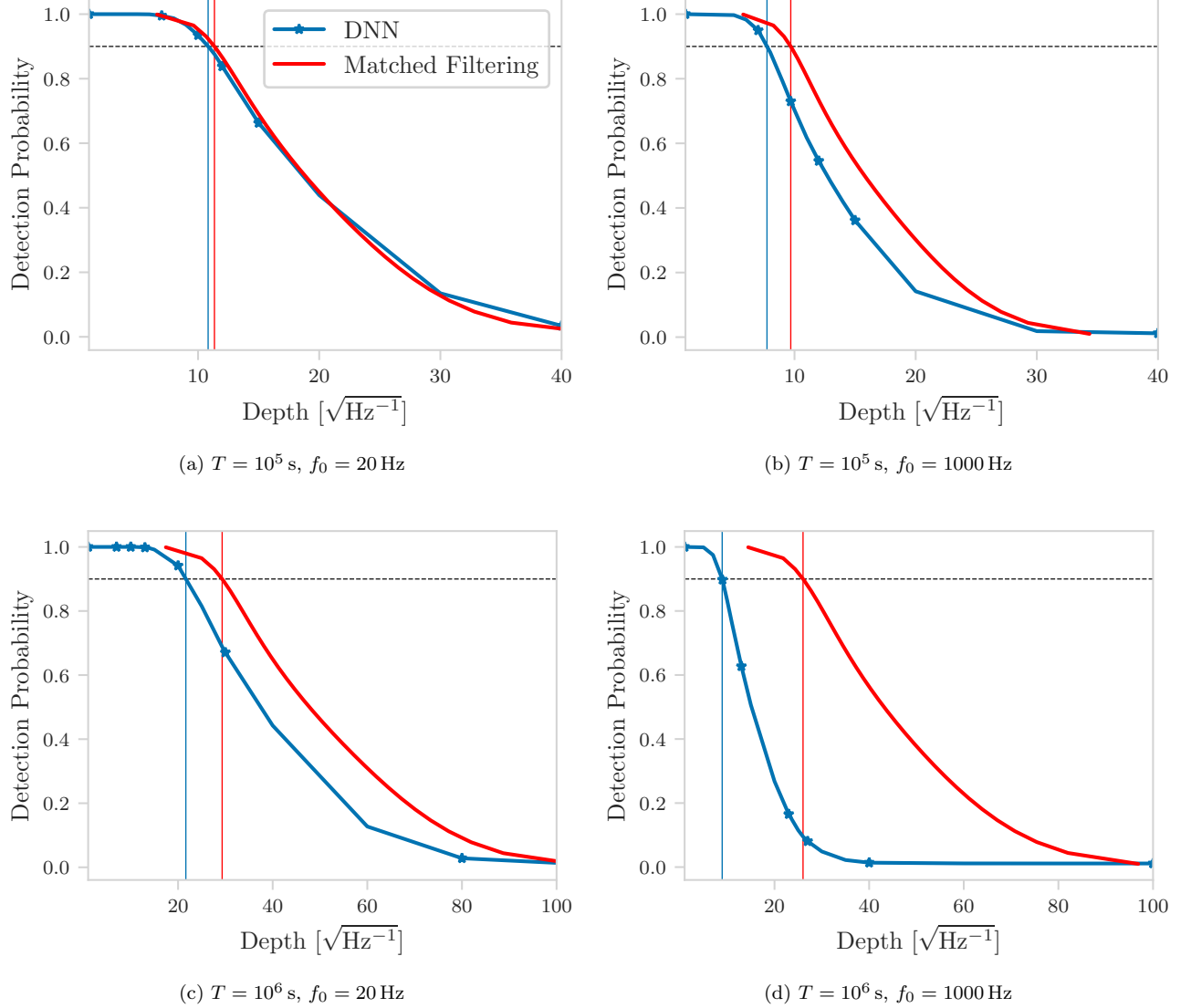


FIG. 8. Detection probability  $p_{\text{det}}$  versus injection depth  $\mathcal{D}$  for networks trained on the respective matched-filtering depth  $\mathcal{D}^{90\%}$  (indicated by the vertical solid line with the diamond at 90%). The second vertical line with the star at 90% gives the sensitivity depth for the DNN at 90% detection probability

sitive overall.

## V. DISCUSSION

In this work we have shown that Deep Learning (DNN) can in principle be used to directly search for CW signals in data, at substantially faster search times than matched filtering. For the hand-optimized network architecture studied here, the DNN detection probability (at fixed false alarm) is found to be somewhat competitive (88% – 73% over the full frequency range) with matched filtering (90%) for short data-spans of  $T \sim 1$  day, while the detection performance falls short (69% – 13%) for a longer data span of  $T \sim 12$  days. On the plus side,

the DNN search shows a surprising ability to extend further in frequency and spindown than it was trained for, and is generally much faster in search performance than matched filtering.

Overall we find that Deep Learning has potential to become a useful CW search tool, but probably a lot more work and effort is required to achieve this. A few immediate ideas we are planning to pursue next in this project are: automated large-scale architecture optimization, training for parameter-estimation in addition to pure classification, extending it to a multi-detector search, and investigating performance on non-Gaussian detector noise.

## ACKNOWLEDGMENTS

We thank Marlin Schäfer, Maria Alessandra Papa and the AEI CW group for helpful comments. We further thank Ruining Zhao for his help in starting this work.

All WEAVE Monte-Carlo simulations and all DNN training and testing were performed on the ATLAS computing cluster of the Albert-Einstein Institute in Hannover. C.M. is supported by the Science and Technology Research Council (grant No. ST/L000946/1).

- 
- [1] D. George and E. A. Huerta, “Deep Neural Networks to Enable Real-time Multimessenger Astrophysics,” *Phys. Rev. D*, **97**, 044039 (2018), arXiv:1701.00008v3 [astro-ph.IM].
  - [2] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, “Matching matched filtering with deep networks in gravitational-wave astronomy,” *Phys. Rev. Lett.* **120**, 141103 (2018), arXiv:1712.06041 [astro-ph.IM].
  - [3] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, *et al.*, “Observation of Gravitational Waves from a Binary Black Hole Merger,” *Phys. Rev. Lett.* **116**, 061102 (2016), arXiv:1602.03837 [gr-qc].
  - [4] B. P. Abbott, R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, V. B. Adya, *et al.*, “GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral,” *Phys. Rev. Lett.* **119**, 161101 (2017), arXiv:1710.05832 [gr-qc].
  - [5] Alexander H. Nitz, Collin Capano, Alex B. Nielsen, Steven Reyes, Rebecca White, Duncan A. Brown, and Badri Krishnan, “1-OGC: The First Open Gravitational-wave Catalog of Binary Mergers from Analysis of Public Advanced LIGO Data,” *ApJ* **872**, 195 (2019), arXiv:1811.01921 [gr-qc].
  - [6] The LIGO Scientific Collaboration and the Virgo Collaboration, “GWTC-1: A Gravitational-Wave Transient Catalog of Compact Binary Mergers Observed by LIGO and Virgo during the First and Second Observing Runs,” arXiv e-prints, arXiv:1811.12907 (2018), arXiv:1811.12907 [astro-ph.HE].
  - [7] J. Aasi *et al.* (LIGO Scientific Collaboration), “Advanced LIGO,” *Class. Quant. Grav.* **32**, 074001 (2015).
  - [8] F. Acernese *et al.* (VIRGO), “Advanced Virgo: a second-generation interferometric gravitational wave detector,” *Class. Quant. Grav.* **32**, 024001 (2015), arXiv:1408.3978 [gr-qc].
  - [9] Reinhard Prix, “Gravitational waves from spinning neutron stars,” in *Neutron Stars and Pulsars* (Springer, 2009) pp. 651–685.
  - [10] P. D. Lasky, “Gravitational Waves from Neutron Stars: A Review,” *PASA* **32**, e034 (2015), arXiv:1508.06643 [astro-ph.HE].
  - [11] K. Riles, “Recent searches for continuous gravitational waves,” *Modern Physics Letters A* **32**, 1730035-685 (2017), arXiv:1712.05897 [gr-qc].
  - [12] Christoph Dreissigacker, Reinhard Prix, and Karl Wette, “Fast and accurate sensitivity estimation for continuous-gravitational-wave searches,” *Phys. Rev. D* **98**, 084058 (2018).
  - [13] R. Prix and B. Krishnan, “Targeted search for continuous gravitational waves: Bayesian versus maximum-likelihood statistics,” *Class. Quant. Grav.* **26**, 204013+ (2009), 0907.2569.
  - [14] P. R. Brady and T. Creighton, “Searching for periodic sources with LIGO. II. Hierarchical searches,” *Phys. Rev. D*, **61**, 082001 (2000), gr-qc/9812014.
  - [15] R. Prix and M. Shaltev, “Search for continuous gravitational waves: Optimal StackSlide method at fixed computing cost,” *Phys. Rev. D*, **85**, 084010 (2012), arXiv:1201.4321 [gr-qc].
  - [16] “Einstein@home project page,” <https://einsteinathome.org/>.
  - [17] V. Dergachev and M. Alessandra Papa, “First loosely coherent all-sky search for periodic gravitational waves in the O1 LIGO data,” arXiv e-prints (2019), arXiv:1902.05530 [gr-qc].
  - [18] J. Ming, M. Alessandra Papa, A. Singh, H.-B. Eggenstein, S. J. Zhu, V. Dergachev, Y.-M. Hu, R. Prix, B. Machenschalk, C. Beer, O. Behnke, and B. Allen, “Results from an Einstein@Home search for continuous gravitational waves from Cassiopeia A, Vela Jr. and G347.3,” arXiv e-prints (2019), arXiv:1903.09119 [gr-qc].
  - [19] The LIGO Scientific Collaboration, the Virgo Collaboration, B. P. Abbott, R. Abbott, T. D. Abbott, S. Abraham, F. Acernese, K. Ackley, C. Adams, R. X. Adhikari, and *et al.*, “All-sky search for continuous gravitational waves from isolated neutron stars using Advanced LIGO O2 data,” arXiv e-prints (2019), arXiv:1903.01901 [astro-ph.HE].
  - [20] Michael A. Nielsen, *Neural Networks and Deep Learning* (Determination Press, 2015).
  - [21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning* (MIT Press, 2016) <http://www.deeplearningbook.org>.
  - [22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *Nature* **521**, 436– (2015).
  - [23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharmashan Kumaran, Thore Graepel, *et al.*, “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” *Science* **362**, 1140 (2018).
  - [24] OpenAI, “Openai five,” <https://blog.openai.com/openai-five/> (2018).
  - [25] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M. Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Yuhuai Wu, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver, “AlphaStar: Mastering the Real-Time Strategy Game

- StarCraft II,” <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/> (2019).
- [26] N. Mukund, S. Abraham, S. Kandhasamy, S. Mitra, and N. S. Philip, “Transient classification in LIGO data using difference boosting neural network,” *Phys. Rev. D* **95**, 104059 (2017), arXiv:1609.07259 [astro-ph.IM].
- [27] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. K. Katsagelos, S. L. Larson, T. K. Lee, C. Lintott, T. B. Littenberg, A. Lundgren, C. Østerlund, J. R. Smith, L. Trouille, and V. Kalogera, “Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science,” *Classical and Quantum Gravity* **34**, 064003 (2017), arXiv:1611.04596 [gr-qc].
- [28] M. Razzano and E. Cuoco, “Image-based deep learning for classification of noise transients in gravitational wave detectors,” *Classical and Quantum Gravity* **35**, 095016 (2018), arXiv:1803.09933 [gr-qc].
- [29] D. George, H. Shen, and E. A. Huerta, “Classification and unsupervised clustering of LIGO data with Deep Transfer Learning,” *Phys. Rev. D* **97**, 101501 (2018), arXiv:1706.07446 [gr-qc].
- [30] S. Vinciguerra, M. Drago, G. A. Prodi, S. Klimenko, C. Lazzaro, V. Necula, F. Salemi, V. Tiwari, M. C. Tringali, and G. Vedovato, “Enhancing the significance of gravitational wave bursts through signal classification,” *Classical and Quantum Gravity* **34**, 094003 (2017), arXiv:1702.03208 [astro-ph.IM].
- [31] P. Astone, P. Cerdá-Durán, I. Di Palma, M. Drago, F. Muciaccia, C. Palomba, and F. Ricci, “New method to observe gravitational waves emitted by core collapse supernovae,” *Phys. Rev. D* **98**, 122002 (2018), arXiv:1812.05363 [astro-ph.IM].
- [32] S. Klimenko, G. Vedovato, M. Drago, F. Salemi, V. Tiwari, G. A. Prodi, C. Lazzaro, K. Ackley, S. Tiwari, C. F. Da Silva, and G. Mitselmakher, “Method for detection and reconstruction of gravitational wave transients with networks of advanced detectors,” *Phys. Rev. D* **93**, 042004 (2016), arXiv:1511.05999 [gr-qc].
- [33] D. George and E. A. Huerta, “Deep Learning for Real-time Gravitational Wave Detection and Parameter Estimation: Results with Advanced LIGO Data,” *Physics Letters B* **778**, 64–70 (2018), arXiv:1711.03121 [gr-qc].
- [34] T. Gebhard, N. Kilbertus, G. Parascandolo, I. Harry, and B. Schölkopf, “Convwave: Searching for gravitational waves with fully convolutional neural nets,” in *Workshop on Deep Learning for Physical Sciences (DLPS) at the 31st Conference on Neural Information Processing Systems (NIPS)* (2017).
- [35] X. Fan, J. Li, X. Li, Y. Zhong, and J. Cao, “Applying deep neural networks to the detection and space parameter estimation of compact binary coalescence with a network of gravitational wave detectors,” *Science China Physics, Mechanics, and Astronomy* **62**, 969512 (2019), arXiv:1811.01380 [astro-ph.IM].
- [36] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, “Convolutional neural networks: a magic bullet for gravitational-wave detection?” arXiv e-prints (2019), arXiv:1904.08693 [astro-ph.IM].
- [37] A. C. Searle, “Monte-Carlo and Bayesian techniques in gravitational wave burst data analysis,” ArXiv e-prints (2008), arXiv:0804.1161 [gr-qc].
- [38] B. Behnke, M. A. Papa, and R. Prix, “Postprocessing methods used in the search for continuous gravitational-wave signals from the Galactic Center,” *Phys. Rev. D* **91**, 064007 (2015), arXiv:1410.5997 [gr-qc].
- [39] K. Wette, S. Walsh, R. Prix, and M. A. Papa, “Weave: a semicoherent search implementation for continuous gravitational waves,” ArXiv e-prints (2018), arXiv:1804.03392 [astro-ph.IM].
- [40] S. Walsh, K. Wette, M. Alessandra Papa, and R. Prix, “Optimising the choice of analysis method for all-sky searches for continuous gravitational waves with Einstein@Home,” arXiv e-prints (2019), arXiv:1901.08998 [astro-ph.IM].
- [41] Piotr Jaranowski, Andrzej Krolak, and Bernard F Schutz, “Data analysis of gravitational-wave signals from spinning neutron stars: The signal and its detection,” *Phys. Rev. D* **58**, 063001 (1998).
- [42] Reinhard Prix, *The  $\mathcal{F}$ -statistic and its implementation in ComputeFstatistic.v2*, Tech. Rep. (2015) (LIGO-T0900149-v5).
- [43] R. Prix, “Template-based searches for gravitational waves: efficient lattice covering of flat parameter spaces,” *Class. Quant. Grav.* **24**, S481 (2007), 0707.0428.
- [44] K. Wette, “Lattice template placement for coherent all-sky searches for gravitational-wave pulsars,” *Phys. Rev. D* **90**, 122010 (2014), arXiv:1410.6882 [gr-qc].
- [45] Jing Ming, Maria Alessandra Papa, Badri Krishnan, Reinhard Prix, Christian Beer, Sylvia J. Zhu, Heinz-Bernd Eggenstein, Oliver Bock, and Bernd Machenschalk, “Optimally setting up directed searches for continuous gravitational waves in Advanced LIGO O1 data,” *Phys. Rev. D* **97**, 024051 (2018).
- [46] Reinhard Prix, *Characterizing timing and memory requirements of the  $\mathcal{F}$ -statistic implementations in LALSuite*, Tech. Rep. (2017) T1600531-v4+.
- [47] Karl Wette, Reinhard Prix, David Keitel, Matthew Pitkin, Christoph Dreissigacker, John T. Whelan, and Paola Leaci, “OctApps: a library of Octave functions for continuous gravitational-wave data analysis,” *Journal of Open Source Software* **3**, 707 (2018).
- [48] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” arXiv e-prints (2015), arXiv:1512.03385 [cs.CV].
- [49] François Chollet *et al.*, “Keras,” <https://keras.io> (2015).
- [50] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” (2015), software available from tensorflow.org.
- [51] LIGO Scientific Collaboration, “LALSuite: FreeSoftware (GPL) tools for data-analysis.” .