Matching Matched Filtering with Deep Networks in Gravitational wave Astronomy

Hunter Gabbard,* Fergus Hayes, Chris Messenger, and Michael Williams
SUPA, School of Physics and Astronomy,
University of Glasgow,
Glasgow G12 8QQ, United Kingdom

(Dated: October 18, 2017)

We report a new method for classifying gravitational-wave (GW) signals from binary black hole (BBH) mergers using a deep convolutional neural network. Using only the raw time series as an input, we are able to distinguish GW signals injected in Gaussian noise amongst instances of purely Gaussian noise time series with (**need figure of merit here**) percent accuracy. We compare our results with the standard method of matched filtering used in Advanced LIGO and find the methods to be comparable.

PACS numbers: May be entered using the `\pacs{#1}` command.

Introduction — The field of gravitational wave astronomy has seen an explosion of binary black hole detections over the past several years [1–3]. These detections were made possible by the Advanced Laser Interferometer Gravitational wave Observatory (aLIGO) detectors, as well as the recent joint detection of GW170814 with Advanced Virgo [4]. Over the coming years many more such observations, including other more exotic sources such as binary neutron star (BNS), intermediate black hole (IMBH), and neutron star black hole (NSBH) mergers, are likely to be observed on a more frequent basis. As such, the need for more efficient search methods will be more pertinent as the detectors increase in *volume* × *time* sensitivity.

The algorithms used to make these detections [5] [**cite gstlal too??**] are computationally expensive to run. Part of the reason being that the methods used by these *search pipelines* are complex, sophisticated processes run over a large parameter space using advanced signal processing techniques. Distinguishing noise from signal in this search pipeline, and others like it, is done using a technique called matched template filtering. Matched template filtering uses a *bank* of template waveforms that spans the astrophysical parameter space [6–17]. We span a large astrophysical parameters space because we do not know *a priori* what the parameters of the gravitational waves in the data are. Because the waveforms of the signals are well modeled, the pipeline uses matched filtering to search for those signals burried in the detector noise. More on how we implement this technique in comparisons with our model will be mentioned later in the methods section of this letter.

We propose that a deep learning algorithm which requires only the raw data time series as input with minimum signal processing would be one alternative search method. This pipeline would be able to be pretrained and then run on real-time detector data with maximum efficiency, as well as in low-latency.

Deep learning is a subset of machine learning which has gained in popularity over the past several years [18–23]. A deep learning algorithm is composed of arrays of processing units, called neurons, which can be anywhere from one to several layers deep (**explain more about what a neuron is?**). Deep learning algorithms typically consist of an input layer, followed by one to several hidden layers and then one to N neurons that output a single value each. This value can then either be used to solve classfication, or regression-like problems. In the case of classification, each output neuron corresponds to the probability that a particular sample is of a certain class.

In our model, we use a variant of a deep learning algorithm called a convolutional neural network (CNN) [24]. CNN layers are composed of five primary variants: input, convolutional, activation, pooling, and fully-connected. Where input holds the raw pixel values of the sample image, the convolutional layer computes the convolution between the kernel and a local region of the input layer volume, activation applies an elementwise activation function leaving the size of the previous layer's output volume unchanged, pooling performs a downsampling operation along the spatial dimensions, and the fully-connected layer computes the class scores using an error function, cross-entropy, defined as

$$f_{\theta'}(\theta) = -\sum_i \theta'_i \log(\theta_i), \tag{1}$$

where θ_i is the predicted probability distribution of class i and θ_i is the true probability for that class [25].

In the following sections we will discuss our choice of network architecture and tuning of it's hyperparameters, compare the results of our network with the widely used GW signal classification technique called matched filtering, and comment on future improvements related to this work.

Methods — In this analysis, in order to make the problem simple, we only distinguish between BBH merger signals injected into a Gaussian noise time series from pure white Gaussian noise time series. The time series for

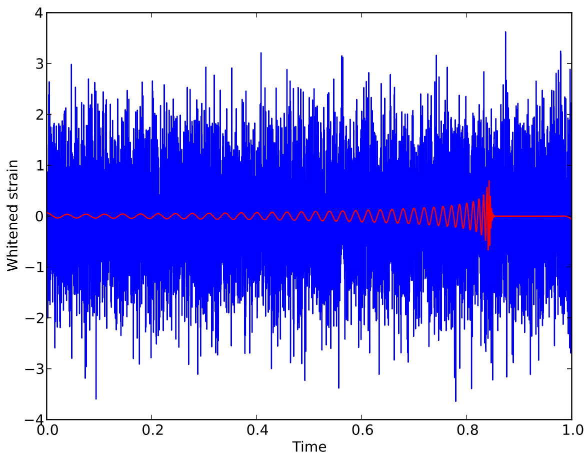

FIG. 1. Description here.

both classes of signals are 1s in duration sampled at 8192 Hz. **say more about how noise and injection are generated using Chris's code.**

For our noise signals we generate a power spectrum density (PSD) that is comparable to aLIGO design sensitivity using a library of gravitational wave data analysis routines called `LALSuite`. That PSD is then converted to an amplitude time series where a random phase shift is given to each spectral component. The inverse real fast fourier transform (IFFT) is then applied and returns a Gaussian time series.

Injections are made using the IMRPhenomD type waveform [26, 27] where the component masses of the waveform range from $5M_{\odot}$ to $100M_{\odot}$, $m_1 > m_2$, and all with zero spin (**not sure if correct?**). Each injection is given a random sky location. The waveforms are then randomly placed within the time series, see figure 1, where the peak of the waveform is within the last 20% of the time series (**perhaps give reason for why this is done**). The waveform is normalised using the integrated signal-to-noise ratio (iSNR), where SNR is defined as

$$\rho_{opt}^2 = 4 \int_0^{\inf} \frac{|h(\hat{f})|^2}{S_n(f)} \mathrm{d}f, \tag{2}$$

where ρ_{opt} is the optimal SNR, $h(\hat{f})$ is the strain amplitude, and $S_n(f)$ is the PSD.

In our runs we used $1,000$ Gaussian noise signals and $1,000$ unique injections with over 25 varrying noise realizations resulting in a total of $50,000$ samples. The samples are then arranged in the form of a 1×8192 pixel sample which is scaled by the GW strain amplitude, $h(t)$, over one color channel (grayscale). (**give ligo definition of optimal SNR**) 70% of these samples are used for training, 15% for validation, and 15% for testing.

In order to achieve the optimal network, multiple sets of hyperparameters are tuned. First, we rescaled the data, but with the existing setup, this did not seem to improve upon the performance. We also attempted applying transfer learning where we used networks trained on successively higher SNR values, though performance benefits were minimal. Network depth was adjusted between 2 to 10 convolutional layers. Our initial data set needed at least 4 convolutional layers. Later data sets with various noise realizations needed fewer convolutional layers to perform comparatively well, but adding more layers still seemed to improved performance. The inclusion of dropout was used within the fully-connected layers as a form of regularization.

For updating our weights and bias parameters (in order to minimize our loss function, $f(\theta)$, (1)) we settled on the nesterov momentum optimization function

$$v_{t+1} = \mu v_t - \epsilon \nabla f(\theta_t + \mu v_t), \tag{3}$$

$$\theta_{t+1} = \theta_t + v_{t+1}, \tag{4}$$

where $\epsilon > 0$ is the learning rate, $\mu \in [0, 1]$ is the momentum coefficient, and $\nabla f(\theta_t)$ is the gradient with respect to the weight vector θ_t. Nesterov momentum was the ideal choice because of its prescient ability to approximate the next position of the weights and bias parameters which therefore gives a rough approximation of their values (**perhaps this is a bit off topic**). Thus the gradient is calculated not with respect to the current parameters, but with respect to the approximate future positions of those parameters [28]. A further detailed description of the neural network architecture used can be found in Table I.

Largely following matched filtering techniques used on the LIGO Open Science Center optimal matched filter page [29] we compare our results to the standard optimal matched filtering process used by aLIGO [30]. Considering the example of one candidate GW signal, we iterate over a comprehensive template bank. The template bank was generated using 8000 randomly sampled mass pairs from the same distribution with no adjustment to assure the parameter space was adequately covered. For each template, we compute the Fast Fourier Transform (FFT) of the data and the template, where the template has been zero padded in order for both to be of the same length. Finally, we multiply the fft'd template and data together and divide by the PSD. An inverse FFT is then applied in order to convert back to the time domain. The output is then normalized so that we have an expected value of 1 for pure Gaussian noise.

Results — After tunning several hyperparameters and then settling on an ideal network format (Table I), we

TABLE I. The optimal network structure (seen below) was determined through multiple tests and tunnings of hyperparameters by means of trial and error. The network consists of 8 convolutional layers, followed by 2 fully-connected layers. Max-pooling is performed on the first, fifth, and eigth layer, whereas dropout is only performed on the two fully-connected layers. Each layer uses an Elu activation function while the last layer uses a Softmax activation function in order to normalize the output values to be between zero and one so as to give a probability value for each class.

	layer 1	layer 2	layer 3	layer 4	layer 5	layer 6	layer 7	layer 8	layer 9	layer 10
Number of Kernals	8	16	16	32	64	64	128	128	64	2
Filter Size	32	16	16	16	8	8	4	4	n/a	n/a
Max Pooling	yes	no	no	no	yes	no	no	yes	no	no
Fully Connected	no	no	no	no	no	no	no	no	yes	yes
Drop out	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
Activation Function	Elu	Elu	Elu	Elu	Elu	Elu	Elu	Elu	Elu	Softmax

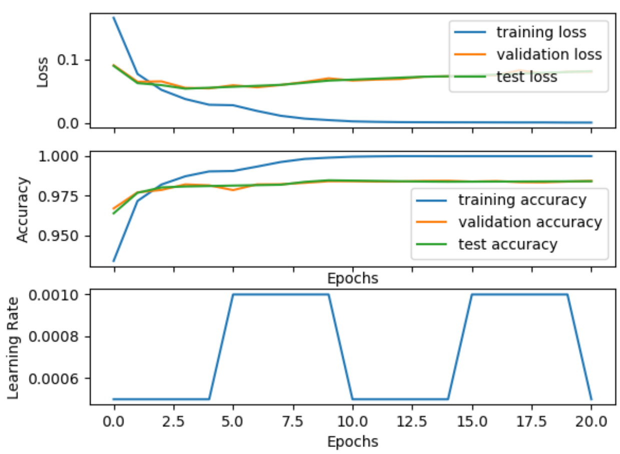

FIG. 2. The loss, accuracy and learning rate plots (shown above) illustrate how the network's performance is defined as a function of the number of training epochs. The goal is to minimize the loss function, which will in turn maximize the accuracy of the classifier. The first initial epochs see an exponential decrease in the loss function and then a slowly falling monotonic curve to follow. This indicates that the longer our network is trained, a limit with respect to the accuracy is approached. In our case, we cyclically adjust the learning rate to oscialte between 5×10^{-4} and 1×10^{-3} at a constant frequency. Studies have shown that this policy of learning rate adjustement (**should replace figure with better run**)

present the results of our classifier on a noise vs. injection sample set. We trained our network using a transfer learning approach whereby we initially trained our network on a sample set of 1000 noise and 1000 injection signals (each with 25 different noise realizations) with an integrated SNR value of 12. We then lowered the integrated SNR value by 2 and (using the same weights from our previous network) we trained our classifier again. This approach seemed to only have a marginal benefit on the overall performance of the classifier.

The confusion matrix in figure 3 shows 99.86% accuracy with triggers at an integrated SNR of 12, 99.64% accuracy with integrated SNR 10, 97.88% at integrated SNR 8, 88.72% at an integrated SNR of 6, 68.34% at integrated SNR 4, and 50.19% accuracy at integrated SNR 2.

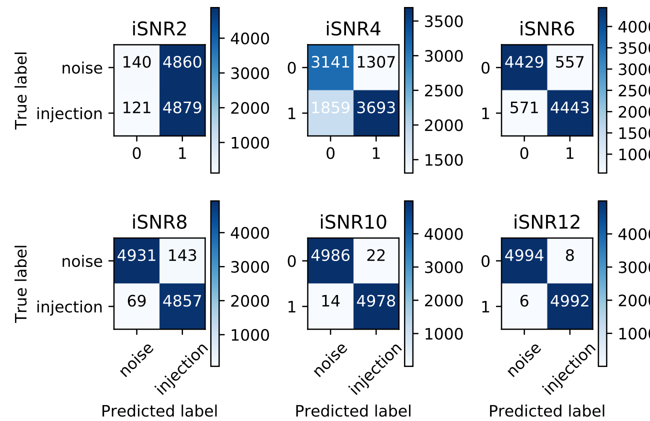

FIG. 3. Confusion matrices for runs from iSNR 2 - iSNR 12. The accuracies for all are listed as follows: 50.19% at iSNR 2, 68.34% at iSNR 4, 88.72% at iSNR 6, 97.88% at iSNR 8, 99.64% at iSNR 10 and 99.86% at iSNR 12. Note that runs with iSNR 2 give results that are equivalent randomized guessing.

In figure 4 we compare our results to that of matched filtering where we use two alternative match filtering methods. The first, using the nominal template bank described in the *Methods* section, whereas the second uses the optimal template for each injection, whereby optimal is defined as the template used to generate that injection. As seen in figure 4 all three methods have equivalent performance proficiency at $\sim$ iSNR > 9, whereas there is a marginal dip in performance in the nominal matched filtering method and our deep learning classifier. It should be noted that our classifier exceeds the performance proficiency of that of the nominal matched filtering method between iSNR 2 and iSNR 4.

We compare the results of all three methods at various injection iSNR values in figure 5. It is not surprising to see that the matched filtering method using the optimal template consistantly performs better than both

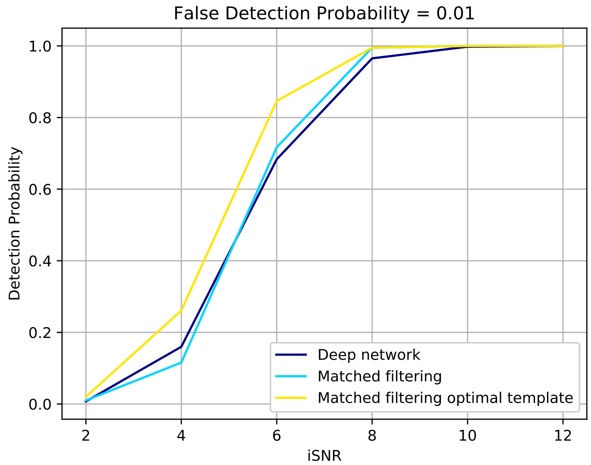

FIG. 4. **Place description here.**

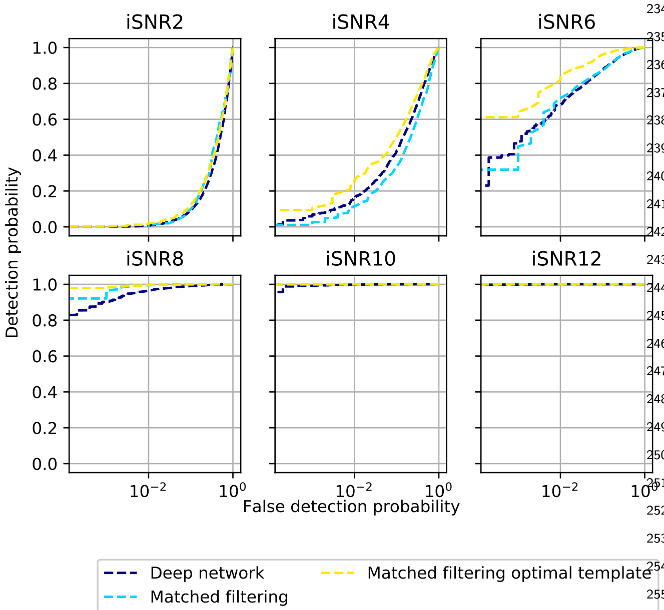

FIG. 5. **Place description here.**

the nominal match filtering method and our deep learning classifier. However, what is considerable is the comparison between the nominal matched filtering and the deep learning classifier detection probability curves. It can clearly be seen that our classifier exceeds the performance of the nominal matched filtering method at iSNR 2, 4 and 6. This is a promising result and certainly merits further investigation.

Conclusions — In conclusion, we demonstrate that deep learning, when applied to a raw data time series, is able to produce equivalent results to matched template filtering. We employ a deep convolutional neural network with carefully chosen hyperparamters and produce an output that returns the class probability value of any given signal. This output could then further be applied as a ranking statistic in the aLIGO CBC search. Although only BBH mergers were used, this method could also easily be applied to other merger types, such as BNS and NSBH signals. The results shown in both figure 4 and figure 5 indicate that deep learning approaches even have the unprecedented potential to entirely exceed matched template filtering.

This work was supported by blah blah blah, something something, important people ...

* Corresponding author: h.gabbard.1@research.gla.ac.uk

[1] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Phys. Rev. Lett. **116**, 061102 (2016).

[2] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Phys. Rev. Lett. **116**, 241103 (2016).

[3] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaboration), Phys. Rev. Lett. **118**, 221101 (2017).

[4] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Phys. Rev. Lett. **119**, 141101 (2017).

[5] S. A. Usman, A. H. Nitz, I. W. Harry, C. M. Biwer, D. A. Brown, M. Cabero, C. D. Capano, T. D. Canton, T. Dent, S. Fairhurst, M. S. Kehl, D. Keppel, B. Krishnan, A. Lenon, A. Lundgren, A. B. Nielsen, L. P. Pekowsky, H. P. Pfeiffer, P. R. Saulson, M. West, and J. L. Willis, Classical and Quantum Gravity **33**, 215004 (2016).

[6] B. S. Sathyaprakash and S. V. Dhurandhar, Phys. Rev. D **44**, 3819 (1991).

[7] S. V. Dhurandhar and B. S. Sathyaprakash, Phys. Rev. D **49**, 1707 (1994).

[8] B. J. Owen, Phys. Rev. D **53**, 6749 (1996).

[9] B. J. Owen and B. S. Sathyaprakash, Phys. Rev. D **60**, 022002 (1999).

[10] S. Babak, R. Balasubramanian, D. Churches, T. Cokelaer, and B. S. Sathyaprakash, Classical and Quantum Gravity **23**, 5477 (2006).

[11] I. W. Harry, B. Allen, and B. S. Sathyaprakash, Phys. Rev. D **80**, 104014 (2009).

[12] D. A. Brown, I. Harry, A. Lundgren, and A. H. Nitz, Phys. Rev. D **86**, 084017 (2012).

[13] P. Ajith, N. Fotopoulos, S. Privitera, A. Neunzert, N. Mazumder, and A. J. Weinstein, Phys. Rev. D **89**, 084041 (2014).

[14] D. Keppel, Phys. Rev. D **87**, 124003 (2013).

[15] D. Keppel, "Metrics for multi-detector template placement in searches for short-duration nonprecessing inspiral gravitational-wave signals," (2013), arXiv:1307.4158.

[16] S. Privitera, S. R. P. Mohapatra, P. Ajith, K. Cannon, N. Fotopoulos, M. A. Frei, C. Hanna, A. J. Weinstein, and J. T. Whelan, Phys. Rev. D **89**, 024003 (2014).

[17] C. Capano, I. Harry, S. Privitera, and A. Buonanno, Phys. Rev. D **93**, 124007 (2016).

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and

K. Q. Weinberger (Curran Associates, Inc., 2012) pp. 1097–1105.

[19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," (2014), arXiv:1406.2661.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," (2014), arXiv:1409.1556.

[21] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," (2014), arXiv:1412.7062.

[22] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," (2013), arXiv:1311.2901.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," (2014), arXiv:1409.4842.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Proceedings of the IEEE **86**, 2278 (1998).

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015), software available from tensorflow.org.

[26] S. Husa, S. Khan, M. Hannam, M. Pürrer, F. Ohme, X. J. Forteza, and A. Bohé, Phys. Rev. D **93**, 044006 (2016).

[27] S. Khan, S. Husa, M. Hannam, F. Ohme, M. Pürrer, X. J. Forteza, and A. Bohé, Phys. Rev. D **93**, 044007 (2016).

[28] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13 (JMLR.org, 2013) pp. III–1139–III–1147.

[29] M. Vallisneri, J. Kanner, R. Williams, A. Weinstein, and B. Stephens, Journal of Physics: Conference Series **610**, 012021 (2015).

[30] B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown, and J. D. E. Creighton, Phys. Rev. D **85**, 122006 (2012).