Simulating the Output of a Gravitational Wave Detector Using Advanced Machine Learning

Joshua Brumpton, Supervisor: Dr Chris Messenger

***Abstract*—The objective of this summer research project was to use machine learning techniques to generate simulated gravitational wave detector output. A normalising flow model was trained with real gravitational wave detector output data and used to generate simulated data. Additionally, a conditional flow model was used to extend the generation of small segments of detector output data into longer sequences of data. Further work from this could focus on improving the performance of the flows used, using alternative flow types and implementing such models to give likelihoods for new data being noise or an unusual event.**

I. Generating with Normalising Flows

THE first direct detection of gravitational waves in 2015 was a significant milestone[1]. The detection not only verified the predictions of Einstein's General Relativity, but opened the possibility to use gravitational wave astronomy as a probe for fundamental physics. The use of machine learning in this field is a popular area of research[2].

For the analysis of gravitational wave data, the noise characteristics of the detector need to be taken into account. An initial choice is to use Gaussian noise modelled from the power spectral density (PSD), a distribution of noise amplitude to the detected frequency. This assumes the detector was operating in conditions that perfectly matched the theoretical conditions which generate the PSD. This is clearly limiting, as there can be multiple, possible one-off, noise contributions which could leave significant artefacts. Another choice is to use real noise data, taken from previous runs, to replace the Gaussian noise. This is more useful as the data will likely contain some of the non-Gaussian noise[3].

This project makes use of generative machine learning to learn the underlying distribution of detector noise. A normalising flow model was trained on real data to do this. The normalising flow takes segments of real data and maps them to a latent space, evaluating the loss and altering the weights to reduce its value. To then generate new simulated segments of detector output data, the latent space is sampled and the flow ran in 'reverse'. The conditional flow used at the end of the project works similarly, but takes a segment of data as a *condition* along with the subsequent values as the corresponding data. Then, when the flow is used to generate data, it is given a segment - which can be real data or generated and then generates data to attach to the end. This can then be looped, taking generated data as the condition for the next segment.

The first steps were to import and pre-process real gravitational wave detector data for training. The data was imported using gwpy, in this case the timeseries strain data was all that was needed. Throughout the project Ligo-Livingston data was used (noise artefacts and characteristics will be different for the different detectors). The data was then normalised so that the strain values lie between 0 and 1, this is required to improve the flow performance. The data was then split into segments, the length of which would be the length of data the flow could generate. If the length of the segment is too long, the network will take much longer to train accurately, and require to import much more data. The flow was then set up, the tunable parameters were mainly the number of transforms, the number of layers per transform, number of neurons per layer, learning rate, weight decay, batch size and dropout probability. The frequency distributions of a large sequence of generations highlighted some issues with the flow. Lower frequencies were often underrepresented in the generated data relative to the original.

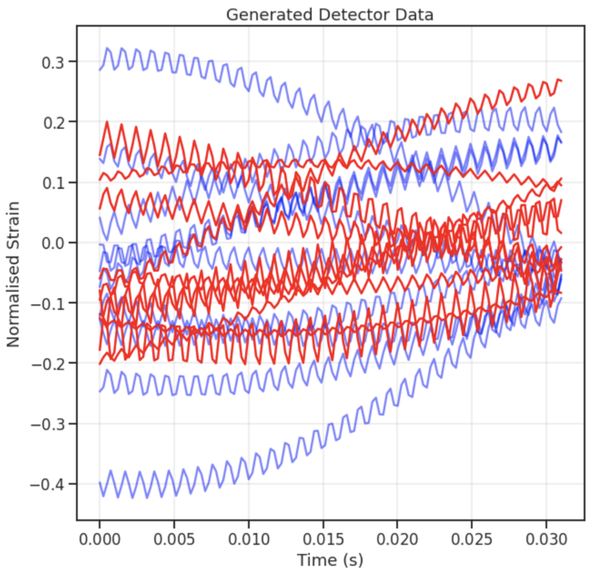

Fig. 1. An example of generated data (red) and original data (blue) for 128 data points.

From figure 1 it is seen that that the generated data has similar characteristics but not quite the same range. This is seen more accurately when analysing the latent space histograms for the flow. It was not possible to get all the latent space histograms to be perfectly normal, there was always a small spread in heights and displacement from 0. The Fourier transform of a large number of generated data segments also highlighted that the flow was not able to learn mainly the low frequencies, see figure 2. The peak at around 60Hz is due to the power lines that operate at that frequency in America. The main portion of time for this stage of the project was spent learning and experimenting with features of the flow. It was found that batch normalization between layers was particularly beneficial[4]. It was useful to spend time altering values for features such as learning rate and dropout probability to assess the sort of affects they would have. While the normalising flow did not work perfectly, it was decided to move on to the use of conditional flows.

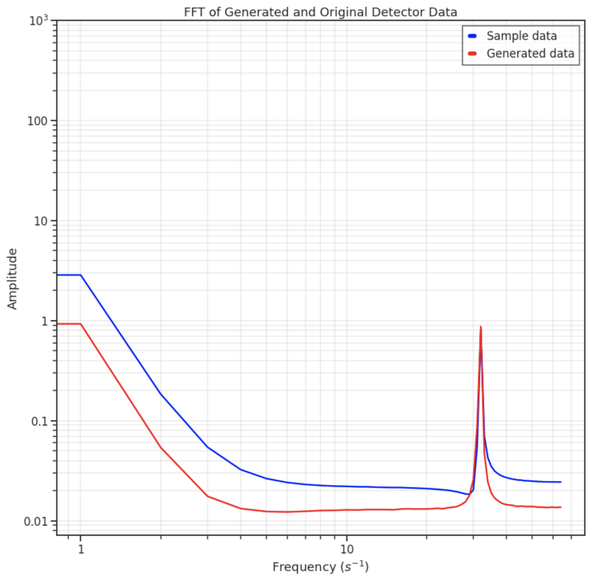

Fig. 2. The fast Fourier transform of generated and original data highlighting the lower amplitudes of frequencies other than 60Hz.

II. Conditional Flows

A conditional flow was used to produce a model that when given a condition, in this case a time series of strain noise data from a gravitational wave detector, generated more data to follow on [5]. For this, the data needs to be split up differently: Rather than just chopping the data into equal size sequences and training the model on these - to reproduce data of the same length - the flow needs to take in specified conditional data. There is now another element to consider, the relative sizes of conditional to non-conditional data. If the conditional data is not long enough then a lot of the low frequency contributions will not be learned adequately, but it can not be arbitrarily long without increasing computation time - you would need more and more data to train on. One possible solution would be to reduce the sample rate earlier in each segment, so that the high frequencies are evident only at the end and not unnecessarily repeated throughout. The conditional flow was trained on much longer pieces of conditional data, of order 1000 data points, which would pick up some of the low frequency behaviour. Example of conditional generations are in figure 3. The generated data carries on with the high frequency behaviour and in the general direction of the conditional data, which is due to lower frequency contributions. However, when run iteratively this low frequency contribution is lost very quickly. The conditional data for the next iteration was altered to include the newly generated data, essentially shifting the conditional data along the time axis - this was then repeated. This was found to not work as intended. The flow had learnt the high frequency portion well but not the lower frequencies so that at each iteration, the low frequency

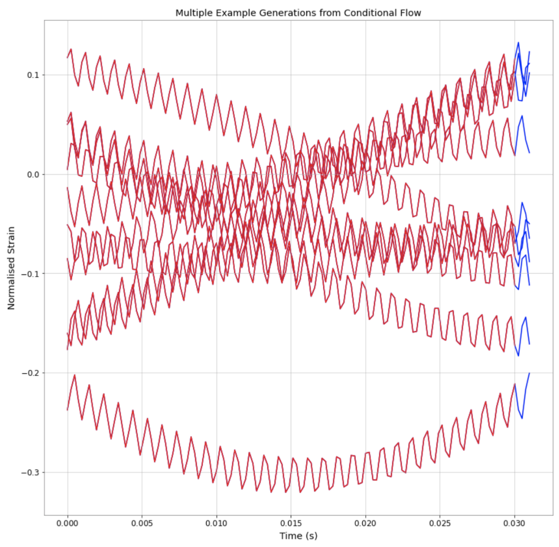

Fig. 3. 10 examples of the conditional flow using 124 conditional data points (red) to generate the next 4 data points (blue).

behaviour was not contributing as much as it should. Over many iterations the conditional data would become flatter meaning that eventually all low frequency behaviour had diminished. An example of this is shown in figure 4. One solution would be to reduce the sampling rate of the conditional data further back in time along the conditional segment. This would mean that the high frequency behaviour is encapsulated at the end and the rest of the segment is smoothed out, only representing the lower frequencies. There was not enough time at the end of the project to implement this. If this were to work then much longer time series data could be generated and the flow could then be used as a discriminator, computing the likelihoods of some real detected data being a consequence of the previous conditional segment - which would highlight significant events or unusual noise artefacts.

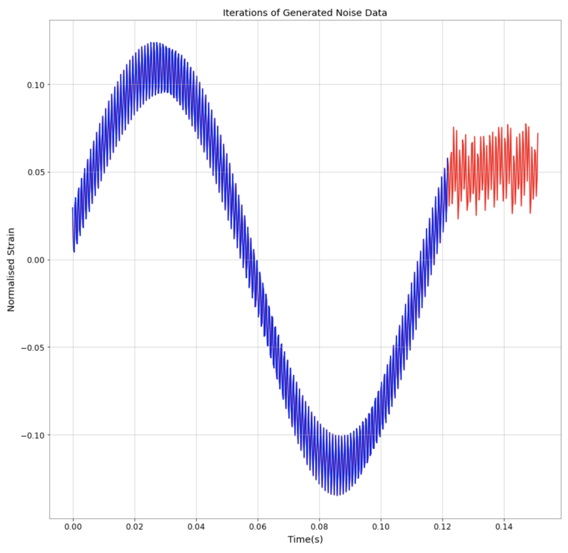

Fig. 4. An example of the conditional flow used iteratively, taking 500 conditional and 12 non-conditional flow 10 times - i.e generating 120 more data points (red) from initial real data (blue).

References

[1] B. P. Abbott, R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, *et al.*, "Observation of gravitational waves from a binary black hole merger," *Physical review letters*, vol. 116, no. 6, p. 061 102, 2016.

[2] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, *Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy*, 2020. arXiv: 1909.06296 [astro-ph.IM].

[3] K. Chatziioannou, C.-J. Haster, T. B. Littenberg, W. M. Farr, S. Ghonge, M. Millhouse, J. A. Clark, and N. Cornish, "Noise spectral estimation methods and their impact on gravitational wave measurement of compact binary mergers," *Phys. Rev. D*, vol. 100, p. 104 004, 10 Nov. 2019. DOI: 10.1103/PhysRevD.100.104004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.100.104004.

[4] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" In *Proceedings of the 32nd international conference on neural information processing systems*, 2018, pp. 2488–2498.

[5] C. Winkler, D. E. Worrall, E. Hoogeboom, and M. Welling, "Learning likelihoods with conditional normalizing flows," *CoRR*, vol. abs/1912.00042, 2019. arXiv: 1912.00042. [Online]. Available: http://arxiv.org/abs/1912.00042.