Massachusetts Institute of Technology

Applying Time Shift Symmetries in Normalizing Flows for Likelihood Free Inference

Maanas Goel[1]
Advisors: Dr. Deep Chatterjee[2], Dr. Erik Katsavounidis[3]

LIGO Laboratory and Kavli Institute for Astrophysics and Space Research,
Massachusetts Institute of Technology
185 Albany Street, Cambridge, Massachusetts 02139, USA

Scientific Research Internship
June - August 2023

[1]maanasg@mit.edu, maanasg@seas.upenn.edu
[2]deep1018@mit.edu
[3]kats@ligo.mit.edu

Contents

1 Abstract

Likelihood free inference using normalizing flows is a popular method for parameter estimation today. However, making likelihood free inferences when there are symmetries, like time translation, in the data is difficult. It also requires more computation time and parameters. Considering the parameter estimation problem in un-modeled gravitational-wave signals as an example; this can be a hurdle for situations where frequency, duration, sky-location are more important parameters compared to the arrival time in a certain segment of data. In this project, we have built a normalizing flow by providing it a time translation symmetry informed embedding network. We demonstrate proof of concept using a simpler model system - a damped simple harmonic oscillator started at different times. We show that when using a pre-trained symmetry informed embedding network, we get faster convergence using a smaller model. Our symmetry embedded neural network can be extended to other relevant problems in gravitational-wave parameter estimation using likelihood-free inference.

2 Introduction to LIGO and Gravitational Waves

The Laser Interferometer Gravitational-Wave Observatory (LIGO) [1], detects gravitational waves predicted by Einstein's General Theory of Relativity. It serves to explore the fundamental physics of gravity, and promotes the development of gravitational wave science as a means for new astronomical discoveries.[4] Using laser interferometry, LIGO measures extremely small ripples in space-time caused by passing gravitational waves produced from massive cosmic events such as the collisions of neutron stars, black holes, and supernovae. LIGO made its first gravitational waves detection in 2015 [2]; the gravitational wave was generated by a pair of merging black holes that are 1.3 billion light years away.[5] Studying gravitational wave astronomy will also answer some of physics's greatest questions: "How do black holes form? Is General Relativity the correct description of gravity? How does matter act under the extremes of temperature and pressure in neutron stars and supernovae?"[6] (Additional information in Appendix.)

LIGO's interferometers consist of two 4-kilometer-scale gravitational wave detectors that work together to detect gravitational waves. As shown in Figure 1, one detector is located in Hanford, Washington, and the other is in Livingston, Louisiana.[2] A single laser beam is split at the point where the two arms of the interferometer intersect to determine the relative lengths of the arms. Half of the laser light travels into one arm, while the other half is reflected into the other arm. Pendulum suspended mirrors are placed at the ends of the arms and at the intersection of the beams. The laser light then bounces back and forth between these mirrors and both the laser lights interfere with each other at the intersection of the arms constructively or destructively. The relative length changes between the two arms detected by light patterns reveals the origins of the gravitational wave source.

Figure 1: **Left:** LIGO detector in Livingston, LA (https://www.ligo.caltech.edu/image/ligo20150731c). **Right:** LIGO detector in Hanford, WA (https://www.ligo.caltech.edu/image/ligo20150731e).

Astrophysical sources like compact binary coalescence (CBC) involving inspiral and merger of compact stellar remnants, have well-modeled theoretical waveform predictions. Weiner matched-filtering techniques have contributed to high-confidence detections of binary-black-hole mergers. However, other gravitational sources with un-modeled or unknown waveforms – such as core-collapse supernovae, magnetar-induced neutron star glitches, and cosmic string cusps – present challenges.[7]

[4]Learn more at https://www.ligo.org/about.php
[5]Learn more at https://www.ligo.caltech.edu/page/about
[6]Learn more at https://www.ligo.org/science.php
[7]Learn more at https://arxiv.org/pdf/1511.05955.pdf

Thus, modeled searches have been detected however, un-modeled signals have not yet been detected. The LIGO detector has shown capabilities of detecting several historical waves. As an example, if the initial wave burst from Supernova 1987 A contained 10^{-4} solar masses of energy or more, it could have been detected by the LIGO detector. This energy level is achievable if the collapsing stellar core was rotating rapidly enough to generate a 0.5 millisecond pulsar. Additionally, the LIGO detector can detect low-frequency radiation in bursts from supernovae with rapidly rotating cores. With the LIGO detector, this is possible in galaxies that are as far as the Virgo cluster of galaxies.[8]

With the LIGO detector, identifying un-modeled transient sources, such as bursts, at high statistical confidence remains a challenge due to the noise component. There is the stationary Gaussian noise, as well as the newly revealed instrumental noise known as glitches. These glitches are challenging to distinguish from astrophysical signals. To address this, Bayesian statistics is used to differentiate between astrophysical signals and glitches and Gaussian noise.[9]

With the increasing rate of detection of the instruments, performing fast inference on the detected signals is becoming progressively computationally challenging. Traditional methods of inference like stochastic sampling (Markov chain Monte Carlo (MCMC) or nested sampling) can be expensive for a real-time characterization of un-modeled gravitational waves. Likelihood-free inference using machine-learning techniques, capable of being accelerated on dedicated hardware, show promise in this direction. In this project, we explore the use of likelihood-free techniques using normalizing flows for parameter estimation of un-modeled gravitational waves signals. We optimize the traditional training scheme of a normalizing flow by pre-training a symmetry aware embedding network for the data to encode the time translated instances of the data. We show that this prescription converges faster compared to a model that is not aware of such symmetries keeping total number of model parameters constant. While we use simplified models in this project, it is a proof of concept for application in real-world application like the parameter estimation of un-modeled gravitational-wave signals using Sine-Gaussian signals, which represent generic burst morphology.

[8]Learn more at https://dcc.ligo.org/public/0065/M890001/003/M890001-03%20edited.pdf
[9]Learn more at https://dcc.ligo.org/public/0119/T1500239/006/Enia_final.pdf

3 Normalizing Flows

Normalizing flows is a powerful method for constructing flexible probability distributions over continuous random variables. The main idea is to apply a transformation T of a random vector u, sampled from a base distribution $p_u(u)$ to represent the target distribution x. Both T and $p_u(u)$ can be uniquely parameterized.

The transformation T must be invertible, and T and T^{-1} must be differentiable. Under these conditions, the density of x, $p_x(x)$ becomes:

$$p_x(x) = p_u(u) \left|\det J_T(u)\right|^{-1}, \quad u = T^{-1}(x) \tag{1}$$

$\det J_T(u)^{-1}$ is the Jacobian determinant which measures the volume change as the transformation maps small neighborhoods from u to x. This allows for flexible reshaping of the base distribution $p_u(u)$ to the target distribution $p_x(x)$. Normalizing flows are powerful because they provide a flexible framework for transforming multidimensional complex transformations. This is due to the fact that invertible and differentiable transformations are composable. Flow refers to the trajectory that samples from $p_u(u)$ follow through a series of transformations. Normalizing refers to the fact that the inverse flow takes samples from $p_x(x)$ and transforms (normalizies) them into the density $p_u(u)$ (See more in Section 2 of [4]). There are two main components to the flow: the transform and the base distribution. When we refer the training the flow, we train the transform.

Autoregressive flows are one of the most popular flows used today. When we refer to autoregressive flows, it is really the transfrom that has the autoregressive property. This flow transforms any given distribution, $p_x(x)$, into a uniform distribution. These flows are implemented by specifying a transformer, τ, and a conditioner, c (See more in Section 3 of [4]). The expressivity of autoregressive flows relies on the flexibility of both the transformer and the conditioner. Different combinations of transformers and conditioners can be used to optimize transformations. In fact, autoregressive flows are universial approximators. Some popular transformers include affine transformers, which are simple and tractable but have limited expressivity. Meanwhile, masked conditioners are more flexible and efficient in evaluation but are more expensive to invert.

Through experimentation with different combinations of transformers and conditioners, we have found Masked Affine Autoregressive Transform as the most optimal flow. Affine transformers are location-scale transformations, where each dimension of the output is a linear combination with scale and shift factors. In masked conditioners, a single neural network is used to compute all conditioning vectors. Specific connections in the network are masked out, allowing for efficient parallel evaluation. This makes masked autoregressive flows universal approximators and computationally efficient in forward computation (See more in Section 3 of [4]).

4 Damped Harmonic Oscillator

We now focus our attention to learning symmetries of the problem, in particular time-translation using a physical model system – the damped simple harmonic oscillator. This serves as a toy problem to the Sine-Gaussian parameter estimation problem. The Damped Harmonic Oscillator is similar to the Sine-Gaussian as both of these problems involve time-shifted data. The goal remains to eliminate the time shift variation as an additional parameter, and focus on the intrinsic parameters of the system like natural frequency and damping coefficient.

The Damped Harmonic Oscillator infers on ω_0, the natural frequency, and β, the damping coefficient. The equation for the Damped Harmonic Oscillator is a second order differential equation:

$$\ddot{x}(t) + 2\beta\omega_0\dot{x}(t) + \omega_0^2 x(t) = 0. \tag{2}$$

Here, x is the displacement over some time t. The solution for this equation is,

$$x(t) = e^{-\beta\omega_0(t-\text{shift})} \cdot \cos\left(\sqrt{1-\beta^2} \cdot \omega_0 \cdot (t - \text{shift})\right). \tag{3}$$

Using the equation above, we showcase in Figure 2 Left signal and data for a Damped SHO, with particular ω and β values as an example. Furthermore, in Figure 2 Right, we also show the time shifted signal plots for the same ω and β value combination. The time shifted signals are therefore the same curve; they are just shifted along the time axis.

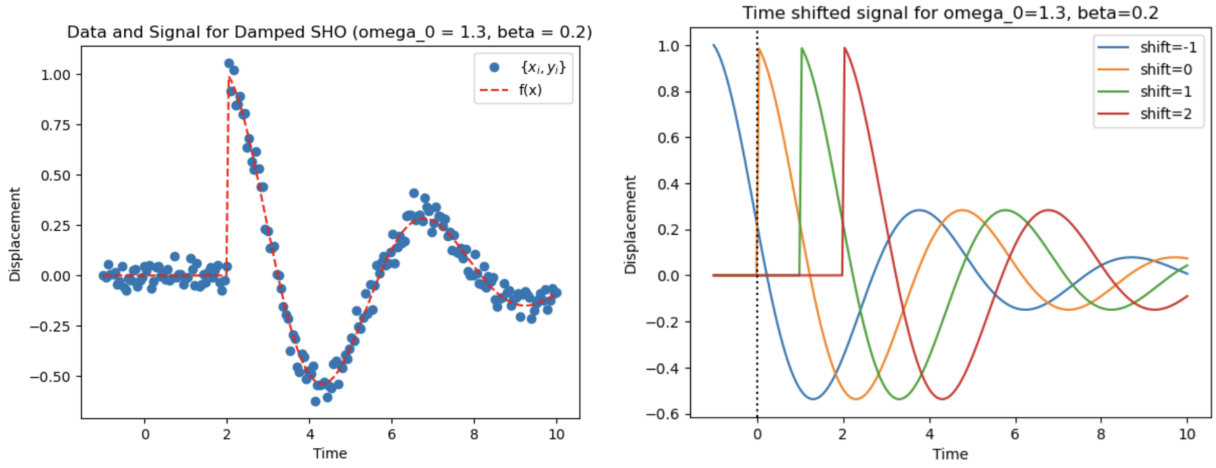

Figure 2: **Left:** Data and Signal for Damped SHO with $\omega_0 = 1.3, \beta = 0.2$ **Right:** Time shifted signals for $omega_0 = 1.3, beta = 0.2$

The time shifted data, shown in Figure 3, is composed of all the time shifted data points for a particular ω and β value combination. All these time shifted data points will be inputs to our neural network, similarity embedding, and the similarity embedding will map these points down to cluster of points in 3D space (as shown in Figure 4 Left). Each cluster will contain points with the same Damped SHO curve (same ω and β values) but will include all the different time shifts. This is beneficial as it eliminates time variation. For this Damped Harmonic Oscillator, we have selected priors, probabilistic distribution, for the parameters. The priors are as follows: ω between 0.1 and 2, β between 0 and 0.5, and time shifts between -4 and 4. The similarity embedding maps down the data points down in a 3D space, and Figure 4 Left shows the sets of 2D projections.

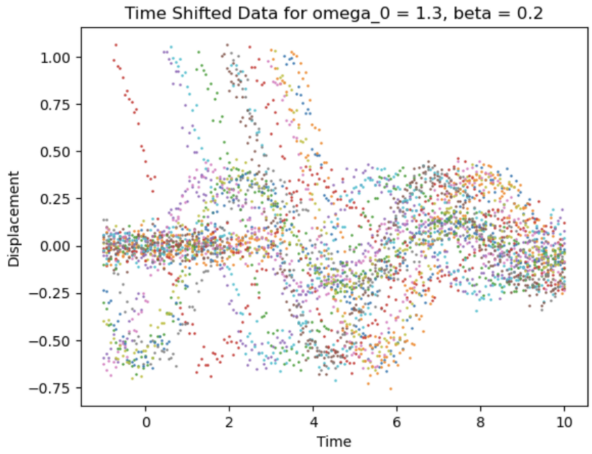

Figure 3: Time shifted data for $omega_0 = 1.3, beta = 0.2$

The reason we use the similarity embedding is because otherwise all the time shifted data is treated as *unique*. Our objective is to infer on the signal's intrinsic parameters such as its frequency, duration, or skylocation consider gravitational-wave source. Therefore, we aim to create a neural network that is agnostic to time shifts. If we simply train a normalizing flow, we would need a bigger model and more data to train. This requires computationally more expensive. Therefore, we used a symmetry informed normalizing flow, which requires a smaller model and less data. It is also computationally faster.

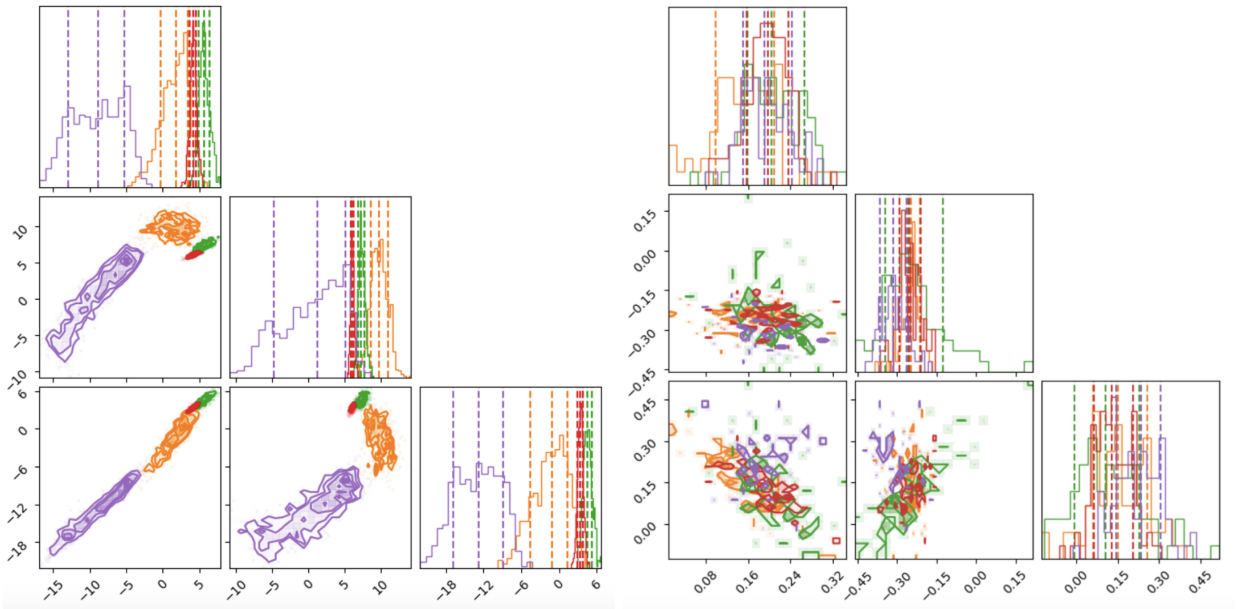

Figure 4: **Left:** Similarity Embedding Corner Plot **Right:** Without Similarity Embedding Corner Plot

Since we have determined that the similarity embedding gives more efficient results faster, we implement it before training our flow. Will now see how we implemented this similarity embedding. For our similarity embedding loss, we computed a VICReg Loss, which stands for Variance-Invariance-Covariance Regular-

ization [3]. The VICReg loss computes a weighted sum of three components: variance, which makes the cluster points dense; invariance, which makes the same data that have the same ω and β values closer, which concurrently separates the clusters with different ω and β value combinations; covariance, which makes the clusters more circular in shape. After experimenting with different combinations of coefficients for these three loss components, we optimized the coefficients to maximize the loss. For the Damped Harmonic Oscillator, the coefficients for the VICReg loss were 50, 1, 50 for invariance, covariance, variance respectively. Furthermore, we utilized the Convolutional 2D Residual Net as we have observed that it performs better for time translation symmetries compared to simply the Residual Net. The Residual Net, we have observed, works better for propagating gradients.

Consider Figure 4 Right, which shows the 2D projections of the 3D subspace that does not use the similarity embedding neural network. In that plot, there is no clear separation of the different data points. All the points are clustered into one big cluster. More importantly, time shifts is still an extra parameter present. Therefore, as shown by Figure 4 Left, using the similarity embedding provides better clusters for time shifted data and better separation between different combinations of ω and β values.

With our similarity embedding neural network, we performed two cases to train our flow. In the first case, we trained the flow using the trained weights of the similarity embedding, and in the second case, we trained the flow without training the similarity embedding. In the second case, the flow had the same embedding network architecture, however this was not pre-trained using the VICReg loss like the first case. In the first case, we froze the weights from the similarity embedding training. Then, we passed those weights through the Masked Affine Autoregressive Transform. In the second case, we did not train the similarity embedding. Therefore, we did not freeze the parameters. We let the training of the normalizing flow change the weights of the similarity embedding by allowing it to use its default weights and biases.

For both the first and second case, we had 30,000 simulations. Furthermore, we split our train set size, validation set size, and test set size in a 80%, 10%, 10% ratio of the number of simulations. We used 400 as the training and validation batch sizes. For training the similarity embedding for the first case, we used a kernel size of 5, 1 block, 50 input channels, 10 hidden channels, 1 output channel for the Convolutional 2D Residual Net. For the optimizer, we set $5e^{-3}$ as the learning rate. We used three schedulers: ConstantLR, OneCycleLR, ExponentialLR. We switched from the first scheduler to the second scheduler after 5 epochs, and from the second scheduler to the third scheduler after 20 epochs. We had a total of 4936 trainable parameters, and we trained the similarity embedding for 30 epochs.

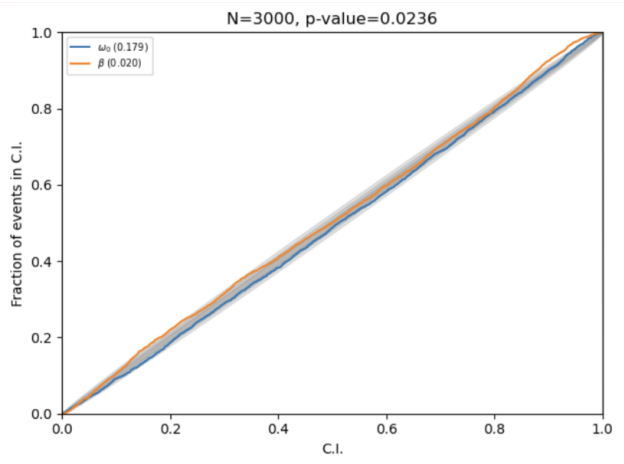

Figure 5: PP Plot

After our training of the flow using the two cases, we have shown the results in Figure 5. In the results we see the pp plot for the flow with the similarity embedding. The diagonal shape of the pp plot shows the accuracy of the training. Both the orange and blue lines (which are the true values for β and ω respectively) lie in the range of the pp plot. The pp plot shows the accuracy of results for 3000 examples. Additionally Figure 6 Left shows the posterior widths of one of the 3000 cases for the flow trained with the similarity embedding. The orange line represents the true value and it lies within the confidence intervals for most cases. We see that compared to the posterior widths of one of the cases of the flow that is not trained with the similarity embedding, in Figure 6 Right, the posterior widths in Figure 6 Left are narrower than the posterior widths in Figure 6 Right. This concludes that the flow with the similarity embedding gives more accurate results than the flow without the similarity embedding. It is important to note that in both cases have the same number of parameters, hyperparameters, learning rates, optimizers, training strategies, and were trained for the same number of epochs. The only difference is in incorporating the similarity embedding.

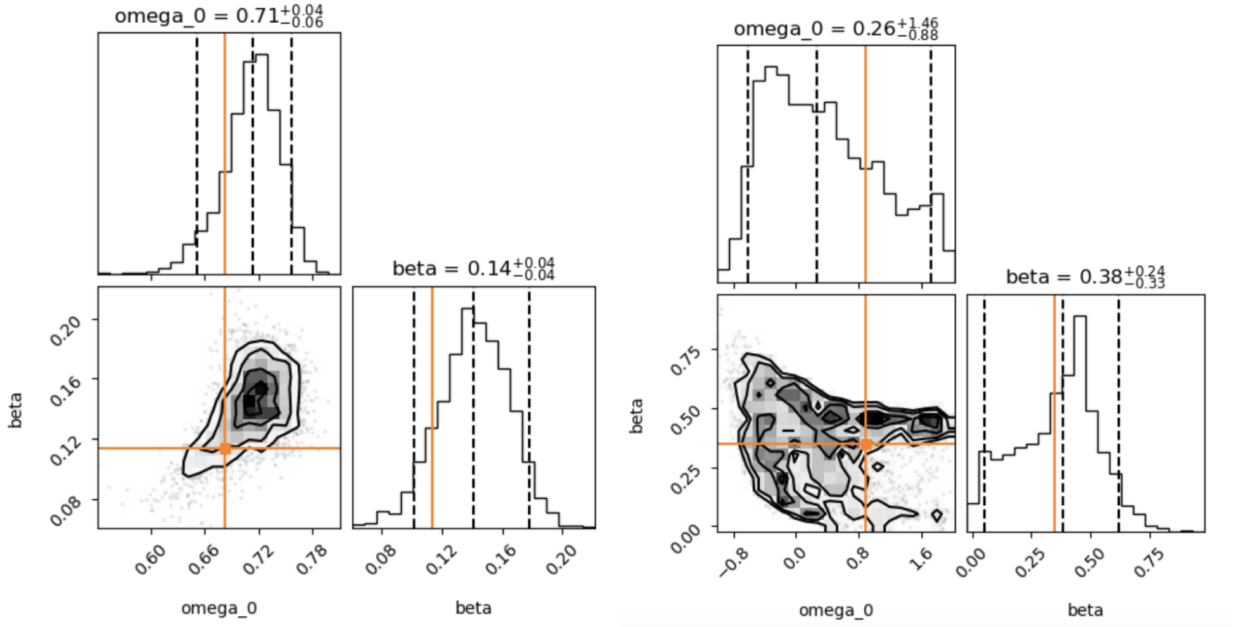

Figure 6: **Left:** Posterior Width example with Similarity Embedding **Right:** Posterior Width example without Similarity Embedding

In addition to having the better accuracy, there is also greater efficiency while using the similarity embedding. Figure 7 shows the comparison of the loss curves with and without using the similarity embedding. The green line loss is with the similarity embedding while the red line loss is without the similarity embedding. It is evident that the losses dive deeper with the similarity embedding, meaning there is more convergence. This concludes that there are more accurate and efficient results that are computationally less demanding with the similarity embedding. Since we have the same number of parameters in both cases, another important conclusion we can draw is that we can get the same results with a smaller number of parameters with the similarity embedding as compared to the results without the similarity embedding. Furthermore, if we do not use the similarity embedding, we would need a bigger model and more parameters which would take more time to train. With the smaller number of parameters, we can train with a smaller model which will require less data to train. It will also train faster, it would require less GPU resources, and it would run more efficiently.

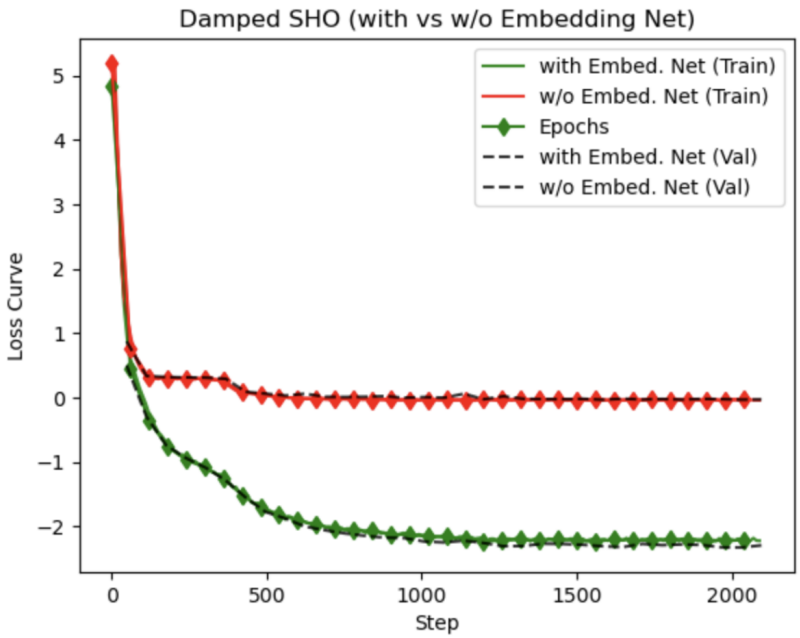

Figure 7: Efficiency of Similarity Embedding in Damped Harmonic Oscillator

5 Sine-Gaussian

Applying the conclusions from the Damped Harmonic Oscillator, we implemented the similarity embedding to LIGO's Sine-Gaussian parameter estimation problem. We have seen that the time translation symmetry informed embedding network improves the training time as well as requires less computational resources.

The Sine-Gaussian waveform is described by:

$$h_{\mathrm{SG}}(t+t_0) = h_0 \cdot \sin(2\pi f_0 t) \cdot e^{-\frac{t^2}{\tau^2}}.^{10} \tag{4}$$

Sine-Gaussian infers on the frequency f_0 similar to the Damped Harmonic Oscillator. The Sine-Gaussian contains $e^{-\frac{t^2}{\tau^2}}$. This term is a Gaussian function that decays as t increases. The exponential factor with the squared term in the denominator controls how fast the Gaussian decays. The Sine-Gaussian also infers on τ (tau), which measures the width of the Gaussian curve. Similar to the Damped Harmonic Oscillator, we showcase in Figure 8 Left signal and data for a Sine-Gaussian, with particular ω and τ values as an example. Furthermore, in Figure 8 Right, we also show the time shifted signal plots for the same ω and τ value combination. The time shifted signals are therefore the same curve; they are just shifted along the time axis.

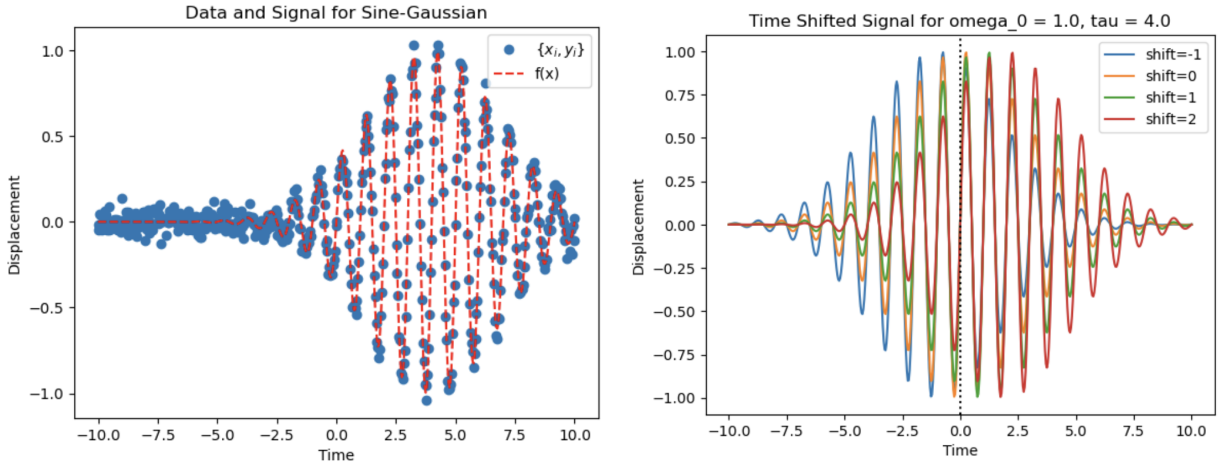

Figure 8: **Left:** Data and Signal for Sine-Gaussian **Right:** Time shifted signals

The time shifted data, shown in Figure 9, is composed of all the time shifted data points for a particular ω and τ value combination. All these time shifted data points were inputs to the similarity embedding neural network, and the similarity embedding mapped these points down to cluster of points in 3D space (as shown in Figure 10 Left). Each cluster contains points with the same Sine-Gaussian curve (same ω and τ values) but will include all the different time shifts. This is beneficial as it eliminates time variation. For this Sine-Gaussian, we have selected priors, probabilistic distribution, for the parameters. The priors are as follows: ω between 0.1 and 1, τ between 0 and 4, and time shifts between -2 and 2. The similarity embedding maps down the data points down in a 3D space, and Figure 10 Left shows the sets of 2D projections.

Consider Figure 10 Right, which shows the 2D projections of the 3D subspace that does not use the similarity embedding neural network. In that plot, there is no clear separation of the different data points. All the points are clustered into one big cluster. More importantly, time shifts is still an extra parameter present. Therefore, as shown by Figure 10 Left, using the similarity embedding provides better clusters for time

[10]Learn more at https://dcc-llo.ligo.org/public/0027/T040055/000/T040055-00.pdf

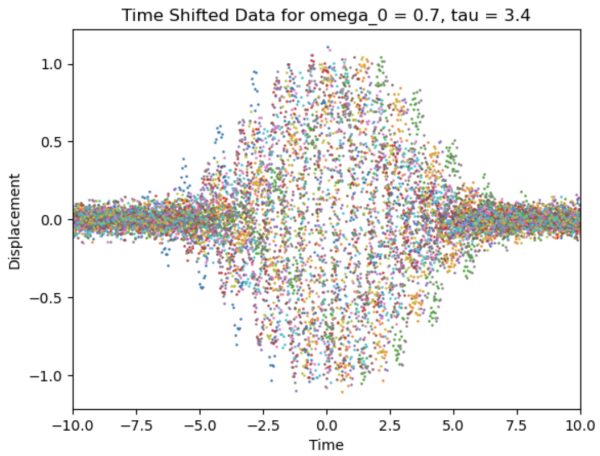

Figure 9: Sine-Gaussian time shifted data

shifted data even for the Sine-Gaussian problem.

With our similarity embedding neural network, we performed two cases to train our flow. In the first case, we trained the flow using the trained weights of the similarity embedding, and in the second case, we trained the flow without training the similarity embedding. In the second case, the flow had the same embedding network architecture, however this was not pre-trained using the VICReg loss like the first case. In the first case, we froze the weights from the similarity embedding training. Then, we passed those weights through the Masked Affine Autoregressive Transform. In the second case, we did not train the similarity embedding. Therefore, we did not freeze the parameters. We let the training of the normalizing flow change the weights of the similarity embedding by allowing it to use its default weights and biases.

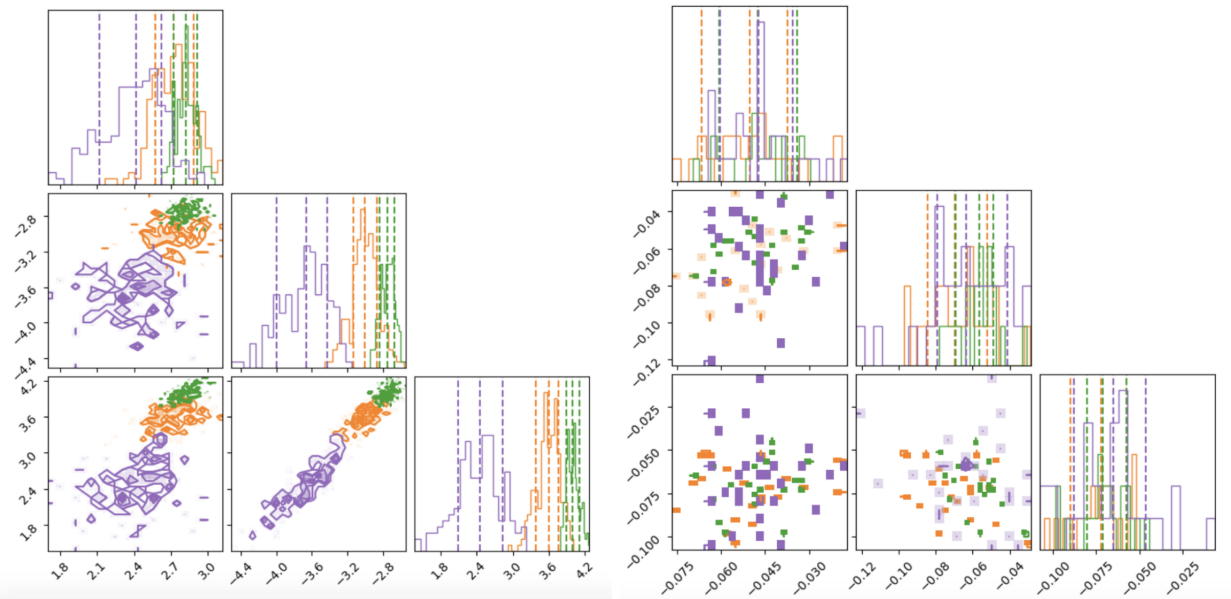

Figure 10: **Left:** Similarity Embedding Corner Plot **Right:** Without Similarity Embedding Corner Plot

For both the first and second case, we had 30,000 simulations. Furthermore, we split our train set size, validation set size, and test set size in a 80%, 10%, 10% ratio of the number of simulations. We used 400 as the training and validation batch sizes. For training the similarity embedding for the first case, we used a kernel size of 5, 1 block, 50 input channels, 10 hidden channels, 1 output channel for the Convolutional 2D Residual Net. For the optimizer, we set $5e^{-3}$ as the learning rate. We used three schedulers: ConstantLR, OneCycleLR, ExponentialLR. We switched from the first scheduler to the second scheduler after 5 epochs, and from the second scheduler to the third scheduler after 20 epochs. We had a total of 5836 trainable parameters, and we trained the similarity embedding for 100 epochs.

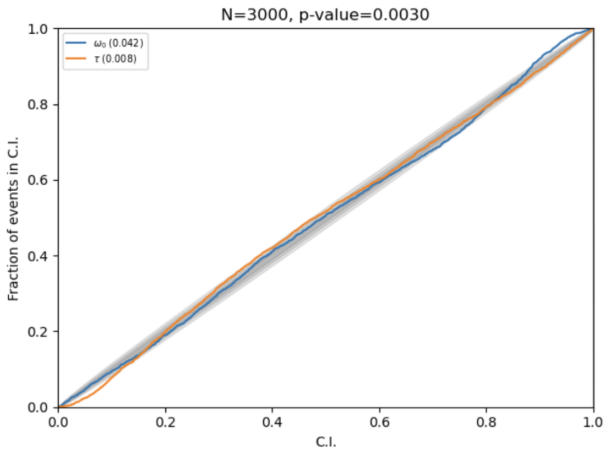

Figure 11: PP Plot with Similarity Embedding

We have shown the training results for the Sine-Gaussian in Figure 11. In the results we see the pp plot for the flow with the similarity embedding. The diagonal shape of the pp plot shows the accuracy of the training. Both the orange and blue lines (which are the true values for τ and ω respectively) lie in the range of the pp plot. The pp plot shows the accuracy of results for 3000 examples. Additionally Figure 12 Left shows the posterior widths of one of the 3000 cases for the flow trained with the similarity embedding. The orange line represents the true value and it lies within the confidence intervals for most cases. We see that compared to the posterior widths of one of the cases of the flow that is not trained with the similarity embedding, in Figure 12 Right, the posterior widths in Figure 12 Left are narrower than the posterior widths in Figure 12 Right. This concludes that the flow with the similarity embedding gives more accurate results than the flow without the similarity embedding. It is important to note that in both cases have the same number of parameters, hyperparameters, learning rates, optimizers, training strategies, and were trained for the same number of epochs. The only difference is in incorporating the similarity embedding.

In addition to having the better accuracy, there is also greater efficiency while using the similarity embedding. Figure 13 shows the comparison of the loss curves with and without using the similarity embedding. Similar to the Damped Harmonic Oscillator, it is evident that the losses dive deeper with the similarity embedding. This concludes that there are more accurate and efficient results that are computationally less demanding with the similarity embedding. For the Sine-Gaussian as well, we can get the same results with a smaller number of parameters with the similarity embedding as compared to the results without the similarity embedding. With the smaller number of parameters, we can train with a smaller model which will require less data to train. This network is trained in such a way that it does not matter where the signal lies within a time segment. It will still be able to detect the signal and provide parameters such as frequency, quality, and skylocation. The advantage is that it requires a smaller network which provides the results faster.

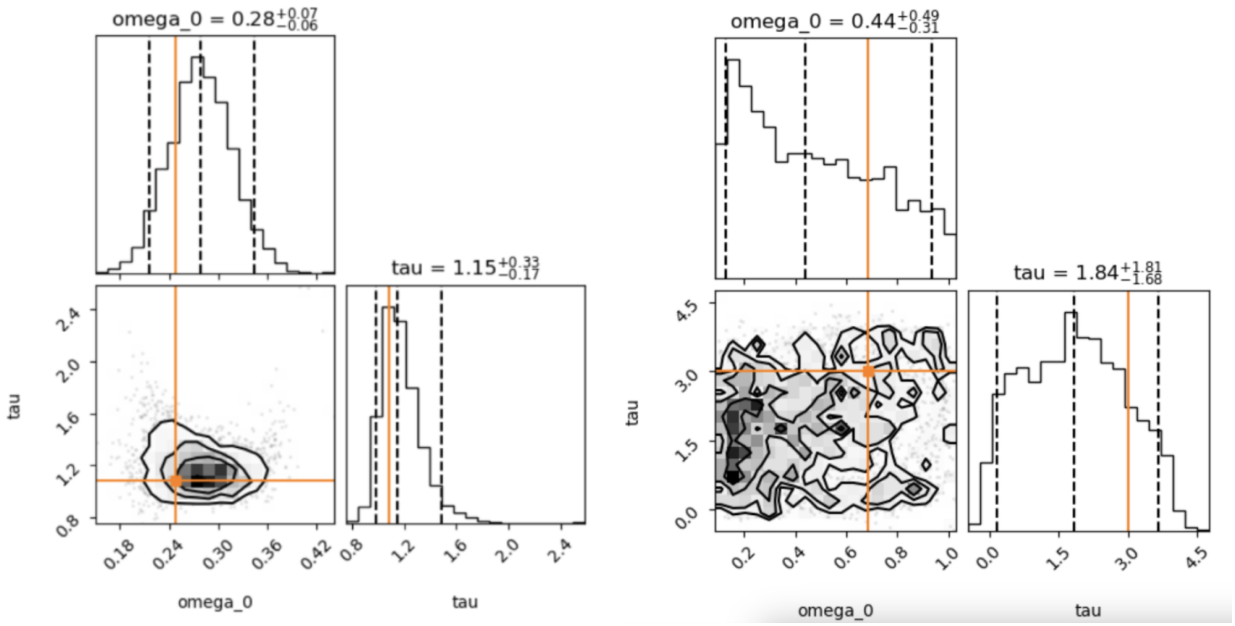

Figure 12: **Left:** Posterior Width example with Similarity Embedding **Right:** Posterior Width example without Similarity Embedding

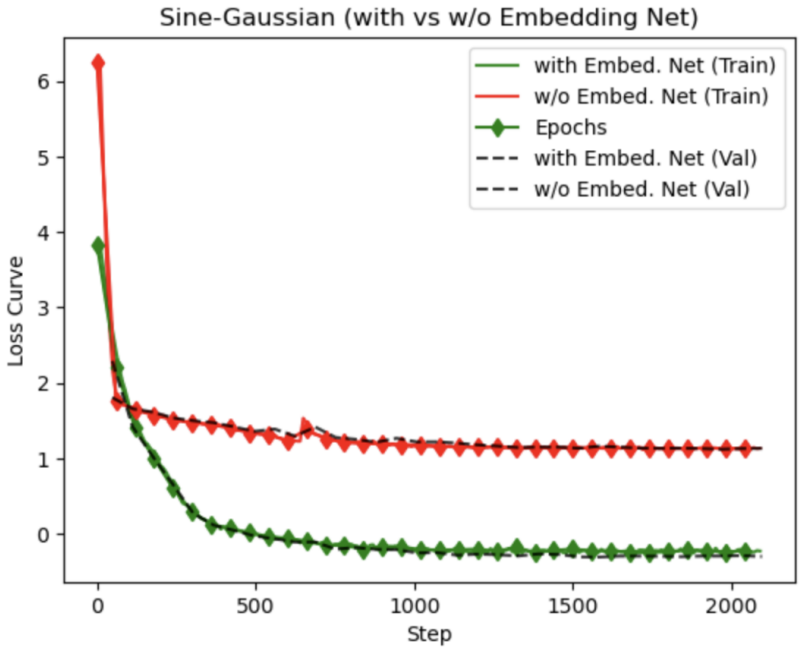

Figure 13: Efficiency of Similarity Embedding in Sine-Gaussian

* * *

6 Appendix

Einstein's Theory of General Relativity suggests that no information can travel faster than the speed of light, including information about the positions of masses in the universe, which travels through gravitational fields. According to General Relativity, any changes in the gravitational field propagate through the universe at the speed of light, producing gravitational waves. Gravitational waves are referred to as "ripples on space-time."[11] Space-time encompasses time along with the three physical dimensions. In this 4-dimensional universe, General Relativity proposes that gravity arises due to the curvature of space-time due to the objects lying in space-time. When this surface is flat, there are no mass-induced 'dents', and therefore objects would move in straight lines. When massive objects dent the surface, other objects are deflected towards the heavy mass due to the curvature, which is known as the gravitational attraction between the two masses. Any changes in the positions of these masses causes ripples on the surface, which is known as changing gravitational fields which produces gravitational waves.[11]

Analogous to electromagnetic waves being created by moving charges, gravitational waves are created from moving masses. Given that gravity is the weakest amongst the fundamental forces (electromagnetism, weak nuclear, and strong nuclear), gravitational waves are incredibly small to detect. As a reference, a strong gravitational wave would produce displacements on the order of $10^{-}18$ meters. This is 10^3 times smaller than the diameter of a proton. Such waves are generated by massive systems with massive accelerations. An example being two black holes orbiting each other to merge into one. These systems are both rare and located several light-years away, thus, the search for gravitational waves aims to detect the minute effects of some of the most energetic astrophysical events in our universe. Gravitational waves are only produced by objects that lack spherical or cylindrical symmetry. Gravitational waves compress objects in one direction and stretch them in the perpendicular direction. To detect these waves, the LIGO detectors have an L-shaped interferometry, which analyzes the interference patterns created when two light sources are combined.[11]

7 Acknowledgements

I would like to sincerely thank Dr. Deep Chatterjee and Dr. Erik Katsavounidis for extending this research opportunity. Extra regards to Dr. Deep Chatterjee for guiding me through the project and always being generous with his time. I would also like to thank other MIT Kavli Institute lab members including Dr. Philip Harris and Ethan Marx. Finally, I want to thank MIT Kavli Institute for welcoming me to their lab as an in-person intern during the summer of 2023. I also acknowledge the support from NSF award 2117997.

8 Contact Author

Maanas Goel
Email: maanasgoel5@gmail.com

[11] Learn more at https://www.ligo.org/science.php

References

[1] J Aasi, B P Abbott, R Abbott, et al. Advanced LIGO. *Classical and Quantum Gravity*, 32(7):074001, mar 2015.

[2] B. P. Abbott, R. Abbott, T. D. Abbott, et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, 116:061102, Feb 2016.

[3] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022.

[4] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2021.