Draft version April 29, 2024
Typeset using LaTeX **twocolumn** style in AASTeX631

Rapid Bayesian Inference of Superluminous Supernovae Using Normalizing Flows

Adam Boesky[1] and Ashley Villar[1]

[1] *Center for Astrophysics | Harvard & Smithsonian, 60 Garden Street, Cambridge, MA 02138-1516, USA*

ABSTRACT

The Vera A. Rubin Observatory will soon begin the Legacy Survey of Space and Time (LSST), in which it will discover over one million supernovae annually. LSST will therefore play a critical role in resolving the intense contention over the progenitors and dynamics of superluminous supernovae (SLSNe)—a very rare supernova class. Conventional methods for classifying and studying supernovae with light curves, however, are far too computationally expensive for the unprecedented volume of data and rate of collection that we expect from LSST. Simulation based inference (SBI) using normalizing flows has been shown to be well-calibrated at rapidly inferring toy supernova parameters compared to traditional methods such as Markov chain Monte Carlo (MCMC). We build upon this work by training a masked autoregressive flow to infer the parameters of SLSNe from highly realistic simulations of Rubin light curves. The pipeline that we propose performs well: it produces accurate, well-calibrated SLSN posteriors for the 5 parameters that we infer in order one-one-thousandth the time. In fact, when compared to MCMC we see that it is better calibrated for a set of test SLSNe, however we are not completely confident in the MCMC implementation that we use. Our study offers a solution to one of the computational challenges that LSST poses, and demonstrates the promise of SBI in the big data era of astrophysics.

1. INTRODUCTION

When stars die, they produce highly energetic explosions called supernovae supernovae (SNe) which come in several different classes. The progenitors and dynamics of one of these classes, superluminous supernovae superluminous supernovae (SLSNe), have been a topic of intense contention since their first observation in 2007 (see Gal-Yam et al. (2009); Pastorello et al. (2010); Chomiuk et al. (2011); Gal-Yam (2012)), largely as a result of high variability in the small set of observations. In particular, their observed peak luminosity and duration span up to an order of magnitude, therefore motivating several different theories for their power sources. A magnetar as the central engine may be the most promising of these theories, as studies have shown its theoretical compatibility with the set of observed properties and lightcurves (Inserra et al. 2013; Nicholl et al. 2014; Nicholl et al. 2018).

The physical distributions of SLSNe has already uncovered valuable physics such as the mass-spin correlation (Blanchard et al. 2020). The number of observed SLSNe to date, however, is relatively low compared to other classes due to their low volumetric rate. Obtaining a more robust understanding of their population distributions will therefore be of principle importance for constraining their many uncertainties in upcoming years.

Fortunately, the Vera C. Rubin observatory will provide an unprecedented volume of supernova (SN) observations from all classes. The observatory is slated to begin its Legacy Survey of Space and Time (LSST) within the next year, in which it will accelerate the discovery rate of SNe from $10,000$ to $\gtrsim 1$ million per year. Legacy Survey of Space and Time (LSST)'s enormous data stream—consisting of roughly ~ 10 million SN alerts nightly—will be so large that it poses serious a logistical challenge: current methods for SN inference such as Markov chain Monte Carlo (MCMC) can take on the order of 10 minutes per light curve, and are therefore prohibitively expensive for application to LSST.

Here, we present a solution to this issue by using simulation-based inference (SBI) to infer SN posteriors with an amortized variational approach. Our method seeks to estimate the posterior $p(\vec{\theta}|\vec{y})$ where $\vec{\theta}$ are the physical parameters of a superluminous supernova (SLSN) and $\vec{y}$ are its light curve observations in the *ugrizy* bands (which are the wavelengths that LSST will observe). Following the methodology in Villar (2022), we estimate the posterior by conducting amortized inference with a masked autoregressive flow (MAF). In particular, we train an MAF on a large set of simulated SLSNe to approximate $p(\vec{\theta}|\vec{y})$ with $q_{\vec{\phi}}(\vec{\phi}|\vec{y})$ where $\vec{\phi}$ are the network's parameters.

We build upon Villar (2022) by (i) simulating SLSNe with a model based on Nicholl et al. (2017) as opposed to the more simplistic core-collapse SNe model from Arnett (1982) and (ii) using cadences, errors, and sampling representative of what is expected from LSST. Astrophysicists have only recently began to use SBI for inference (e.g. Green et al. (2020); Hortúa et al. (2020); Sánchez et al. (2021); Villar (2022)). As the field enters the era of big data, we expect SBI and the use of normalizing flows to only increase in popularity, as the computational speed of circumventing the likelihood function cannot be understated.

2. DATA AND METHODOLOGY

2.1. *The Superluminous Supernova Dataset*

We simulate a set of 66,549 SLSNe using the magnetar spin-down model. The specific model that we use is based on the Bayesian SLSN fitting software `MOSFiT` (Nicholl et al. 2017), which was built upon theory developed in Ostriker & Gunn (1971), Kasen & Bildsten (2010), Chatzopoulos et al. (2012), and Inserra et al. (2013).

Following order, we assume that the magnetar's energy input is that of an ordinary magnetic dipole

$$F_{\rm mag}(t) = \frac{E_{\rm mag}}{t_{\rm mag}}(1 + t/t_{\rm mag})^2 \tag{1}$$

where $E_{\rm mag} = (2.6 \times 10^{52}) P_{\rm s}^{-2} ({\rm M_{NS}}/1.4\ {\rm M_\odot})$ erg is the rotational energy of the magnetar (Lattimer & Schutz 2005) with a spin-down timescale of $t_{\rm mag} = (1.4 \times 10^5)(\frac{P_{\rm s}}{B_\perp})^2 (M_{\rm NS}/1.4\ {\rm M_\odot})^{3/2}$s, and the neutron star is parameterized by a rotational period $P_{\rm s}$, mass $M_{\rm NS}$, and magnetic field perpendicular to the spin axis $B_\perp$. Table 1 lists the distributions that we sample the simulation parameters from, which are the same as those used in Nicholl et al. (2018) and are based on observations by Liu et al. (2017).

The energy output calculated by the spin-down model in Equation 1 is then fed into the analytical luminosity function derived in Arnett (1982)

$$L_{\rm out}(t) = \exp\left(-\frac{t}{t_{\rm diff}}\right)(1 - e^{At^{-2}})\times \int_0^t 2F_{\rm in}(t')\frac{t'}{t_{\rm diff}} e^{\left(\frac{t'}{t_{\rm diff}}\right)^2}\frac{dt'}{t_{\rm diff}} \tag{2}$$

where we assume $F_{\rm in} = F_{\rm mag}$, the diffusion time is $t_{\rm diff} = \sqrt{2\kappa M_{\rm ej}/\beta c v_{\rm ej}}$, $A = 3\kappa_\gamma M_{\rm rj}/4\pi v_{\rm ej}^2$ is the so-called "leakage" parameter, κ is the optical capacity, κ_γ is the opacity for high-energy photons, and the ejecta has mass $M_{\rm ej}$ with velocity $v_{\rm ej}$ at the photosphere.

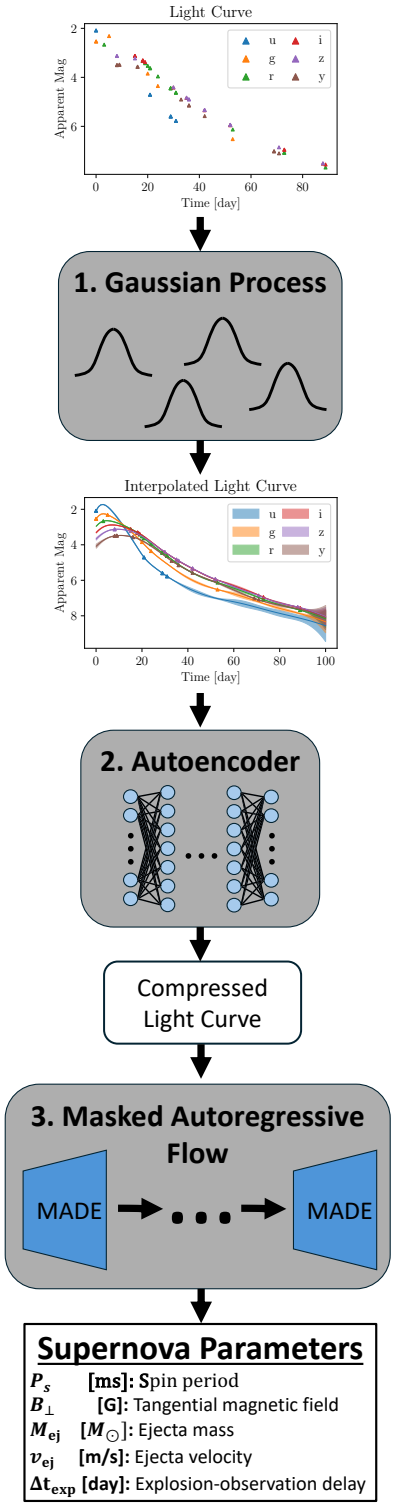

Figure 1. A schematic representation of our pipeline. The pipeline takes in raw light curves in the *ugrizy* wavelengths (1) interpolates their magnitude and errors onto a 100 day long, 1 day cadence grid, (2) compresses them with an autoencoder, and (3) passes the latent lightcurve vector along with the supernova's photometric redshift into a MAF to infer its parameters. The parameter variables, units, and labels are listed in the table at the bottom.

After calculating the intrinsic luminosities of our SNe, we then simulate their observation by LSST. We inject the supernovae uniformly between $z = 0$–2, and then we simulate 100 iterations in which we first check if the supernova is within Rubin's sight. If the supernova will be observed, we time dilate the lightcurves according to their redshifts and then resample them according to LSST's cadence. We then redden the SN light curves according to observed Milky Way extinction curves from Gordon et al. (2009). Finally, we overlay the light curves with noise that we expect to experience in LSST.

Before conducting inference on the lightcurves, we conduct a quality cut on the 66,549 simulated SLSNe. We decide to keep light curves with > 35 photometric measurements that span at least a 70 day period. Only 32,643 light curves satisfy these requirements, halving the initial dataset. We further filter out all light curves with mean signal-to-noise ratios < 3, leaving us with a final collection of 12,525 SLSNe which we randomly split into a train and test set with an 80-20 ratio. When comparing to MCMC, which we describe in Section 2.5, we use a subset of 250 test SLSNe because it takes too long to fit the entire test set.

2.2. *Light Curve Interpolation*

The neural network components of our pipeline have static architectures, making it necessary to convert the light curves to a fixed length vector. We therefore interpolate *ugrizy* values over both wavelength and time onto a fixed, 100-day, 1-day cadence grid of times using a 2D Matern-3/2 Gaussian process from the `George` package (Ambikasaran et al. 2015). We show an example of the original and interpolated light curve in the first and third panels of Figure 1. Nonparametric light curve interpolation with Gaussian processes has been done many times before in the field of SN science (e.g. Boone (2019); Sup (2020); Qu & Sako (2021); Villar (2022)). Because our light curves are more sparsely sampled and atypical than Villar (2022), we use the interpolated errors in inference. Our light curve vectors are therefore length 1,200 where the first 600 values are the 100 interpolated *ugrizy* observations and the last 600 are their corresponding errors.

2.3. *Light Curve Autoencoder Compression*

While the main benefit of SBI is speed, training MAFs can be rather slow when they have many parameters. We therefore use an autoencoder to compress the light curves to a 25-dimensional latent representation so that the input layer of our MAF can have fewer nodes, cheapening its training (25 dimensions for the latent space is chosen arbitrarily). We compare the performance of our pipeline with and without this compression step throughout the paper to see if compressing the light curves impacts performance (see Figure 3 and Figure 5 in Section 3).

Parameter	Distribution	Min	Max	λ	Value
P_s [ms]	Uniform	0.7	20.0		
v_{ej} [10^4km s^{-1}]	Uniform	1.0	30.0		
$B_\perp$ [$10^1$4G]	Log-Uniform	0.01	10		
M_{ej} [$M_\odot$]	Log-Uniform	0.1	100		
Δt_{exp} [day]	Exponential			0.1	
M_{NS} [$M_\odot$]	Fixed				1.4
κ [g cm^{-2}]	Fixed				0.1126
κ_γ [g cm^{-2}]	Fixed				0.1

Table 1. Distributions and values of SLSN parameters used in our simulations and as our inference priors. All values are taken from Nicholl et al. (2018), and the uniform distribution for v_{ej} approximates their truncated Gaussian because it has a narrow range and large variance.

Our autoencoder's architecture is shown in Figure 2, and is comprised of an encoder and decoder which we implement with the `PyTorch` (Paszke et al. 2019) package and tune minimally by hand. The encoder takes in light curves of 600 magnitudes with their corresponding 600 errors and transforms them to the 25-dimensional latent space. It has two hidden layers with rectified linear unit (ReLU) activation functions, followed by a layer that linearly transforms the data to the latent space. The decoder then uses two hidden layers with ReLU activation functions followed by a linear layer to reconstruct the light curves. We train the autoencoder for 1000 epochs using the AdamW algorithm (described in Loshchilov & Hutter (2017)) to optimize the mean squared error (MSE) loss between our original and reconstructed light curves. While learning, we use a batch size of 256, a learning rate of 10^{-3}, and we implement early stopping when validation loss has not improved in over 100 epochs.

A random set of reconstructed light curves that we inspect closely matches their original form, suggesting that our autencoder effectively compresses the light curves while preserving information. All of the simulated SLSNe have a generally similar time evolution, so we expect the network to perform well on the entire set of simulations.

2.4. *Masked Autoregressive Flow Parameter Inference*

The model that we train to conduct SBI is a MAF, which fits posteriors by combining variational inference techniques with a normalizing flow architecture. Typical variational inference methods are restricted to Gaus-

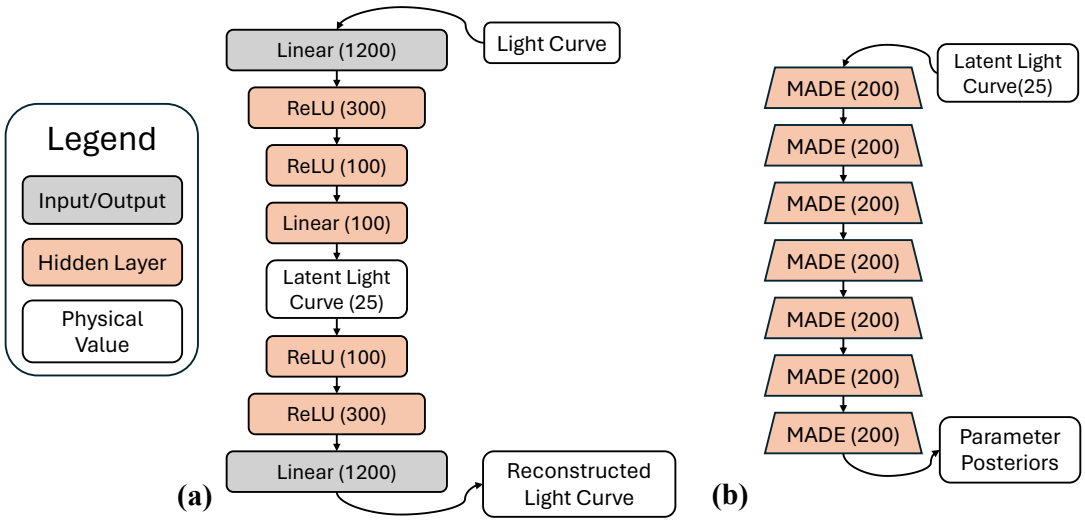

Figure 2. The architectures of **(a)** the autoencoder used to compress our light curves and **(b)** the MAF used to infer the posteriors of our supernova parameters. The number in parenthesis in each layer represents the number of nodes in that given layer.

sian distributions and are therefore insufficient for learning complex posteriors. Normalizing flow architectures help MAFs solve this problem by transforming a Gaussian with a series of non-linear, invertible functions to some estimate $p_{\vec{\phi}}(\vec{\theta}|\vec{y})$ of $p(\vec{\theta}|\vec{y})$. The "masked" aspect of MAFs refers to how networks are connected such that inputs are a function of a subset of previous inputs, therefore making them autoregressive such that outputs can be interpereted as conditional probabilities. Furthermore, their invertibility and use of latent Gaussians endow MAFs with the functionality to sample from $p_{\vec{\phi}}(\vec{\theta}|\vec{y})$ in test time. For more on the theory and methodology behind MAFs, see Papamakarios et al. (2017).

We implement our MAF using the `SBI` module (Tejero-Cantero et al. 2020) and train it similarly to Villar (2022). Our architecture, which we show in Figure 2, is comprised of 7 masked autoencoder for density estimation (MADE) blocks (first presented by Germain et al. (2015)) of 200 nodes each. We train the MAF to infer $P_{\rm s}$, $B_{\perp}$, $M_{\rm ej}$, $v_{\rm ej}$, and $\Delta t_{\rm exp}$ using encoded SLSN light curves along with their redshifts (which we expect to be accurately photometrically constrained in LSST (Graham et al. 2018)). The priors that we adopt for SBI are the same distributions that we sampled from for simulation, and are listed in Table 1. We use the Adam algorithm (Kingma & Ba 2014) to minimize the Kullback-Leibler divergence

$$D_{\rm KL}(p_{\vec{\phi}}(\vec{\theta}|\vec{y}) \parallel p(\vec{\theta}|\vec{y})) = \int p_{\vec{\phi}}(\vec{\theta}|\vec{y}) \log \frac{p_{\vec{\phi}}(\vec{\theta}|\vec{y})}{p(\vec{\theta}|\vec{y})} d\vec{\theta} \quad (3)$$

which describes how similar $p_{\vec{\phi}}(\vec{\theta}|\vec{y})$ is to $p(\vec{\theta}|\vec{y})$. We use a 9-1 training-to-validation split, and find reasonable results with manual hyperparameter tuning, which we describe in detail in Section 3.

To investigate the impact of compressing the light curve before conducting SBI, we train a MAF with the same architecture (except for the input size) as a baseline by inputting the length 1200 light curve vector and its corresponding redshift. We will compare our pipeline to this model as control for how light curve impression impacts performance throughout Section 3.

2.5. *Markov Chain Monte Carlo Parameter Inference*

We fit a random subset of 250 simulated light curves using MCMC to see how SBI compares to conventional methods. To conduct MCMC, we use the `emcee` affine invariant ensemble sampler implementation (Foreman-Mackey et al. 2013). To sample, we use 50 walkers for 6000 steps, at which point we observe that fits are typically converged. We adopt the same priors as for SBI (listed in Table 1), and the log-likelihood function that we use is

$$\mathcal{L}(\vec{y}) = \left(\frac{-1}{2}\right) \sum_i \frac{(y_i - \hat{y}_i)^2}{\sigma^2} + \log(2\pi\sigma_i^2) \quad (4)$$

where i indexes over the light curve y, $\hat{y}$ is the prediction for the current set of parameters, and $\sigma^2 = \sigma_{\rm m}^2 + \sigma_{\rm n}^2$ is the sum of the observed uncertainty and a free parameter used to account for underestimating noise. We use a log-uniform prior between 0 and 2.5 for $\sigma_{\rm n}$, which we expect to be small because our pipeline uses fair uncertainty estimates.

3. RESULTS

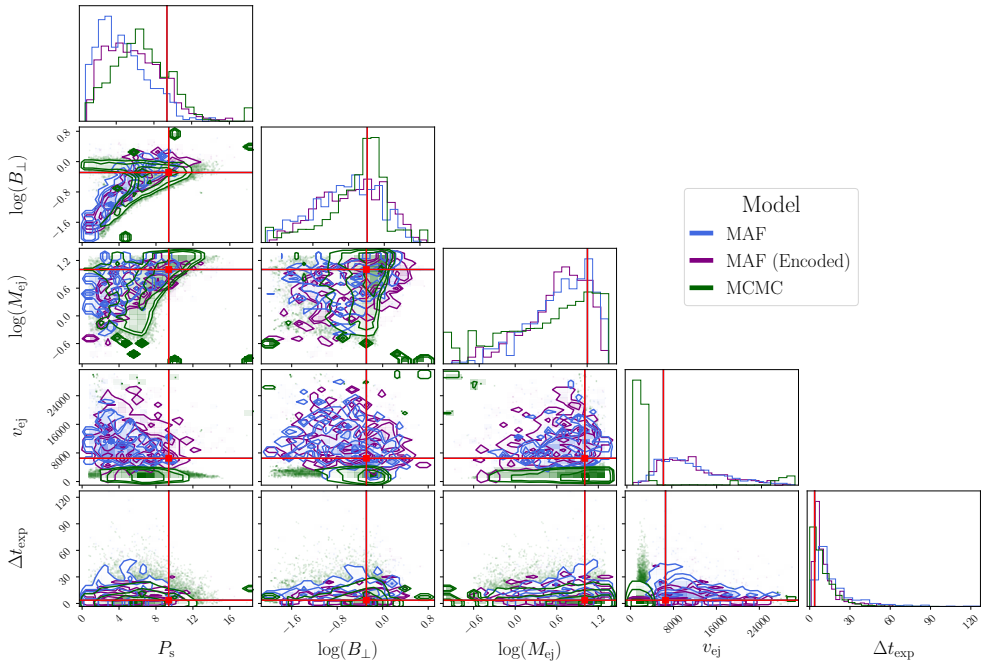

Figure 3. The posteriors of one randomly selected supernova from the test set from MCMC, our proposed pipeline with the light curve encoding step, and our pipeline without the encoding step.

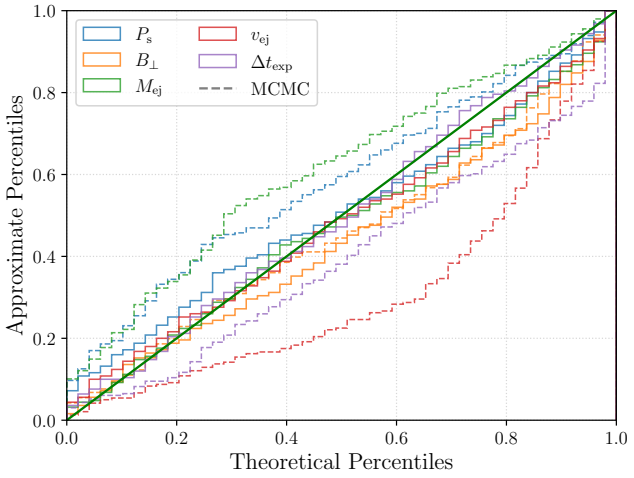

Figure 4. A probability-probability plot for the parameters for the 250 supernova test set used in Figure 5 from the proposed pipeline (solid lines) and MCMC (dashed lines). The lines represent the cumulative distribution of percentile scores for the true parameter values within the one-dimensional, marginalized posteriors. A perfect estimator would product the green diagonal line with slope 1 going through the origin.

We train our pipeline according to the procedure outlined in Section 2, and find that it produces very good posteriors of P_s, $B_\perp$, M_{ej}, v_{ej}, and Δt_{exp} for the test set. These 5 dimensional posteriors are produced by simply drawing from $p_{\vec{\phi}}(\vec{\theta}|\vec{y})$, and in Figure 3 we show an example of the posteriors for one SN in the test set. We also observe that our model performs *better* than MCMC, however we are not completely confident in our MCMC implementation—there may be a bug in our code, or it is possible that the walkers need to explore for than 6000 steps to converge.

In Figure 4 we show a "p-p" (short for "probability-probability") plot for our pipeline and MCMC implementation. These plots communicate how well posterior distributions align with true data by showing the one-to-one relationship between the true and inferred cumulative distribution function (CDF) for a test set. We observe that our pipeline closely follows the green diagonal line for all parameters, which represents a perfectly-calibrated model. On the other hand, MCMC deviates considerably from the diagonal for most of the parameters, straying as far as 0.3. MCMC is particularly poorly calibrated for v_{ej}, but does quite well for $B_\perp$. Poor v_{ej} can also be seen in the example posterior plotted in Figure 3, as the distribution is far from the true value and is relatively narrow. There may be some sort of degeneracy that leads to poor MCMC inference, but this is more likely a bug and our pipeline—the focus of this paper—is able to learn and predict v_{ej} well regardless.

We further explore how the pipeline compares to current methods in Figure 5, as well as how it performs at inferring the test set on the whole. Considering predictions of the whole test set is of particular importance

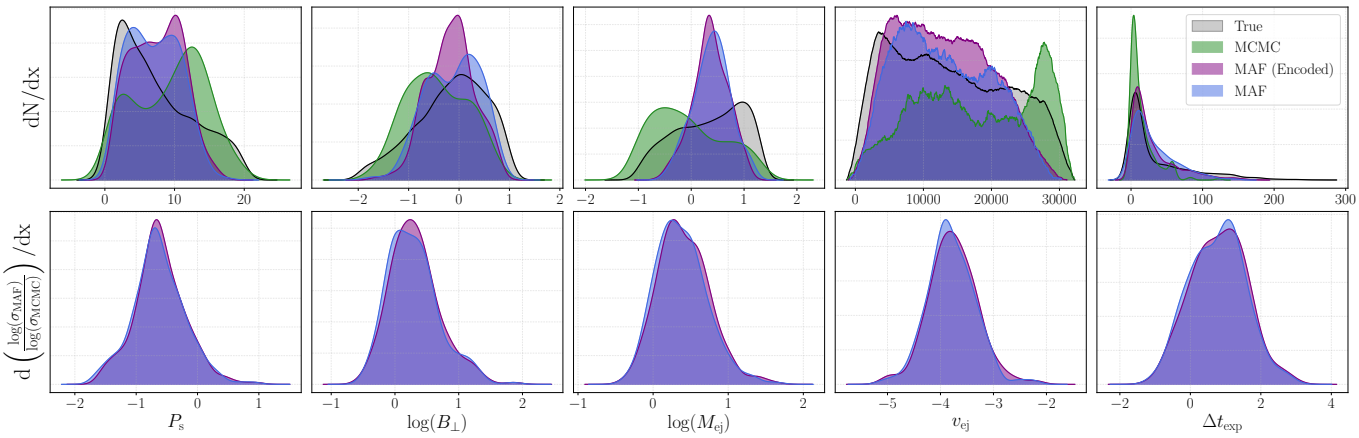

Figure 5. Top: A kernel density estimate (KDE) of median posterior estimates for a 250 SN subset of the test set. We show the median posterior estimate for our proposed pipeline with the encoding step, the pipeline with no encoding step, MCMC. We also include the true distribution of the parameters in gray for reference. **Bottom:** A KDE of the standard deviation of the SBI-inferred posteriors ($\sigma_{\rm MAF}$) divided by the MCMC standard deviations ($\sigma_{\rm MCMC}$). A pipeline that is well calibrated would produce standard deviations close to $\sigma_{\rm MCMC}$. For all KDEs, we use the Silverman's Rule of Thumb (Silverman 1986) for the band width.

for the scope of this study because we aim to create a solution to the *population* inference problem that LSST poses. The pipeline—both with and without the encoding step—produces estimates of the 250 test SLSNe that are significantly different than MCMC for most of the parameters. Again, this is likely a result of the bug, but the predictions for $v_{\rm ej}$ and $M_{\rm ej}$ are the most different while $\Delta t_{\rm exp}$ are actually relatively similar. Our pipeline's posterior also has smaller uncertainties that MCMC for $v_{\rm ej}$ and $P_{\rm s}$, but larger ones for $B_\perp$, $M_{\rm ej}$, and $\Delta T_{\rm exp}$.

The SBI inferences generally reflect the true distribution of the parameters, but seem to favor predictions closer to the mean than at the tails. This phenomenon can be seen by how the SBI distributions in Figure 5 are all narrower than the true distributions and centered near the mean value, and it is especially drastic for $M_{\rm ej}$. While the pipeline is biased in this sense, favoring the population mean is not such a bad issue given that the uncertainties seem to match the data well (as explained before in reference to Figure 4). A bias towards the mean is worth noting for when this pipeline is used to infer the population of real SLSN observations in LSST, and model biases should be explored for future iterations given that SBI is prone to biased inferences. It is also worth investigating if this bias is a result of Gaussian process interpolation, and considering other methods of turning light curves into fixed-length vectors.

Predictions from our pipeline are very similar whether the autoencoder compression step is included or not. In Figure 3 the posteriors from encoded and non-encoded light curves are nearly identical for an example SLSN, and the distributions of 250 test SLSN predictions are very similar in Figure 5. In Figure 5, the uncertainties of the posteriors (bottom row) align very well, but there are a few features in parameter distributions (top row) that are different such as bimodal versus unimodal peaks for $B_\perp$. The autoencoder, therefore, may introduce some small bias into the pipeline, however further tuning it with a grid search could potentially remedy these differences. With the relatively small training set of $10,020$ simulated SLSNe, the autoencoder did not make such a significant difference on the training time of our MAF (training took on the order of one hour with and without the compression), but it may prove beneficial for larger training sets in the future.

Finally, our pipeline is far computationally superior to current methods. Our implementation of MCMC takes roughly 5 minutes to converge per object using one CPU core for all walkers while our pipeline takes about 0.26 seconds per object to draw a 500 sample posterior. Also, the Gaussian process interpolation accounts for the majority of that time, taking ~ 0.16. While our pipeline is in fact slower than the original SBI for SN inference presented in Villar (2022), it is still a factor of 10^3 quicker than MCMC.

4. CONCLUSION

We present a SBI pipeline that uses a MAF to conduct amortized inference on SLSNe. The pipeline first interpolates lightcurves over the *ugrizy* bands and time, then compresses the light curves using an autoencoder, and finally samples an approximation $p_{\vec{\phi}}(\vec{\theta}|\vec{y})$ of $p(\vec{\theta}|\vec{y})$ using an MAF. We train our pipeline on a set of simulated SLSNe using the magnetar model energy input model from Nicholl et al. (2017), and we hand tune all

hyperparameters. To compare our pipeline to current methods, we fit a subset of the test light curves with affine invariant ensemble MCMC. Below, we summarize our findings:

1. Our pipeline produces accurate and well-calibrated posteriors for the test set (Figure 3, Figure 4).

2. The pipeline produces more well-calibrated posteriors than MCMC, the most commonly used method for supernova inference (Figure 4). While this is probably an issue with our implementation and not an issue with MCMC, our pipeline's performance is at the upper limit of what MCMC could achieve regardless.

3. Compressing the data using an autoencoder produces negligible differences on SN inference while marginally reducing training time (Figure 3, Figure 5). Further tuning the autoencoder hyperparameters with a grid search may eliminate inference differences all together.

4. Our model is slightly biased towards the mean of the true distribution and therefore struggles to predict SNe at the tails of the test set (Figure 5). Although the uncertainties on the pipeline's predicted posteriors are ideal, this bias is important to consider when our pipeline is applied to SLSNe observations from LSST because it may impact interpretations made in population-level studies.

5. Inferring SN parameters with our pipeline takes ~ 0.26 seconds per SN, a factor of 10^3 less time than MCMC. The majority of this time is spent interpolating the light curve onto a fixed length grid.

5. FUTURE PROSPECTS

Our pipeline is both well-calibrated and fast, making it a viable replacement for current methods of population-level SN inference in the era of the Vera C. Rubin observatory. Additional perks of our pipeline and SBI methods alike are that they save valuable CPU time and are energy efficient.

Our method will help resolve the many uncertainties about SLSNe being debated in the scholarly conversation, but there are many uncertainties about other classes of SNe that will persist. Luckily, the flexibility of SBI using MAFs lends pipelines like ours the capacity to be trained to infer other SN classes. Future studies should therefore see how the architecture that we present performs on simulated sets of SNe from other classes.

We thank the staff and students of the Harvard Astronomy 205 course for the opportunity to learn about machine learning's applications to Astrophysics.

Software: All code used for this project can be found at this GitHub repository. The following `Python` packages were used: `Matplotlib` (Hunter 2007), PyTorch (`?`), `SBI` (Tejero-Cantero et al. 2020), `NumPy` (Harris et al. 2020), `SciPy` (Virtanen et al. 2020), `IPython/Jupyter` (Perez & Granger 2007; Kluyver et al. 2016), `Astropy` (Collaboration et al. 2018), `KDEpy` (Odland 2018).

REFERENCES

2020, Astrophysical Journal, 905, doi: 10.3847/1538-4357/abc6fd

Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D. W., & O'Neil, M. 2015, IEEE Transactions on Pattern Analysis and Machine Intelligence, 38, 252, doi: 10.1109/TPAMI.2015.2448083

Arnett, W. D. 1982, ApJ, 253, 785, doi: 10.1086/159681

Blanchard, P. K., Berger, E., Nicholl, M., & Villar, V. A. 2020, ApJ, 897, 114, doi: 10.3847/1538-4357/ab9638

Boone, K. 2019, The Astronomical Journal, 158, 257, doi: 10.3847/1538-3881/ab5182

Chatzopoulos, E., Wheeler, J. C., & Vinko, J. 2012, ApJ, 746, 121, doi: 10.1088/0004-637X/746/2/121

Chomiuk, L., Chornock, R., Soderberg, A. M., et al. 2011, ApJ, 743, 114, doi: 10.1088/0004-637X/743/2/114

Collaboration, T. A., Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, The Astronomical Journal, 156, 123, doi: 10.3847/1538-3881/aabc4f

Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, PASP, 125, 306, doi: 10.1086/670067

Gal-Yam, A. 2012, Science, 337, 927, doi: 10.1126/science.1203601

Gal-Yam, A., Mazzali, P., Ofek, E. O., et al. 2009, Nature, 462, 624, doi: 10.1038/nature08579

Germain, M., Gregor, K., Murray, I., & Larochelle, H. 2015, arXiv e-prints, arXiv:1502.03509, doi: 10.48550/arXiv.1502.03509

Gordon, K. D., Cartledge, S., & Clayton, G. C. 2009, ApJ, 705, 1320, doi: 10.1088/0004-637X/705/2/1320

Graham, M. L., Connolly, A. J., Ivezić, Ž., et al. 2018, AJ, 155, 1, doi: 10.3847/1538-3881/aa99d4

Green, S., Simpson, C., & Gair, J. 2020, Physical Review D, 102, doi: 10.1103/PhysRevD.102.104057

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Nature, 585, 357–362, doi: 10.1038/s41586-020-2649-2

Hortúa, H. J., Malagò, L., & Volpi, R. 2020, Machine Learning: Science and Technology, 1, 035014, doi: 10.1088/2632-2153/aba6f1

Hunter, J. D. 2007, Computing in Science and Engineering, 9, 90, doi: 10.1109/MCSE.2007.55

Inserra, C., Smartt, S. J., Jerkstrand, A., et al. 2013, ApJ, 770, 128, doi: 10.1088/0004-637X/770/2/128

Kasen, D., & Bildsten, L. 2010, ApJ, 717, 245, doi: 10.1088/0004-637X/717/1/245

Kingma, D. P., & Ba, J. 2014, CoRR, abs/1412.6980. https://api.semanticscholar.org/CorpusID:6628106

Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, in ELPUB, 87–90

Lattimer, J. M., & Schutz, B. F. 2005, ApJ, 629, 979, doi: 10.1086/431543

Liu, Y.-Q., Modjaz, M., & Bianco, F. B. 2017, ApJ, 845, 85, doi: 10.3847/1538-4357/aa7f74

Loshchilov, I., & Hutter, F. 2017, arXiv e-prints, arXiv:1711.05101, doi: 10.48550/arXiv.1711.05101

Nicholl, M., Guillochon, J., & Berger, E. 2017, The Astrophysical Journal, 850, 55, doi: 10.3847/1538-4357/aa9334

Nicholl, M., Smartt, S. J., Jerkstrand, A., et al. 2014, MNRAS, 444, 2096, doi: 10.1093/mnras/stu1579

Nicholl, M., Blanchard, P. K., Berger, E., et al. 2018, The Astrophysical Journal Letters, 866, L24

Odland, T. 2018, tommyod/KDEpy: Kernel Density Estimation in Python, v0.9.10, Zenodo, doi: 10.5281/zenodo.2392268

Ostriker, J. P., & Gunn, J. E. 1971, ApJL, 164, L95, doi: 10.1086/180699

Papamakarios, G., Pavlakou, T., & Murray, I. 2017, arXiv e-prints, arXiv:1705.07057, doi: 10.48550/arXiv.1705.07057

Pastorello, A., Smartt, S. J., Botticella, M. T., et al. 2010, ApJL, 724, L16, doi: 10.1088/2041-8205/724/1/L16

Paszke, A., Gross, S., Massa, F., et al. 2019, arXiv e-prints, arXiv:1912.01703, doi: 10.48550/arXiv.1912.01703

Perez, F., & Granger, B. E. 2007, Computing in Science and Engineering, 9, 21, doi: 10.1109/MCSE.2007.53

Qu, H., & Sako, M. 2021, Photometric Classification of Early-Time Supernova Lightcurves with SCONE

Sánchez, A., Cabrera, G., Huijse, P., & Förster, F. 2021. https://api.semanticscholar.org/CorpusID:245961782

Silverman, B. W. 1986, Density Estimation for Statistics and Data Analysis (Chapman and Hall/CRC)

Tejero-Cantero, A., Boelts, J., Deistler, M., et al. 2020, Journal of Open Source Software, 5, 2505, doi: 10.21105/joss.02505

Villar, V. A. 2022, arXiv e-prints, arXiv:2211.04480, doi: 10.48550/arXiv.2211.04480

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: 10.1038/s41592-019-0686-2