Multivariate Modeling of Quasar Variability with an Attention-based Variational Autoencoder

MATTHEW WAYNE LOWERY

Multivariate Modeling of Quasar Variability with an Attention-based Variational Autoencoder

Matthew Lowery

A Thesis presented for the degree of
Master of Science by Research

Department of Physics
Durham University
United Kingdom
July 2023

Abstract

This thesis applied HeTVAE [1], an attention-based VAE neural network capable of multivariate modeling of time series, to a dataset of several thousand multi-band AGN light curves from ZTF and was one of the first attempts to use a neural network to harness the stochastic light curves in their multivariate form. Whereas standard models of AGN variability make prior assumptions, HeTVAE uses no prior knowledge and is able to learn the data distribution in a regularized latent space, reading semantic information via its up-to-date self-supervised training regimen. We have successfully created a dataset class for preprocessing the irregular multivariate time series and in order to interface with the quasi-off-the-shelf network more conveniently. Also, we have trained several different model iterations using one, two or all three of the filter dimensions from ZTF on Durham's NCC compute cluster, while configuring useful hyper parameter choices to work robustly for the astronomical dataset. In the network's training, we employed the Adam optimizer with a reduce-on-plateau learning rate schedule and a KL-annealing schedule optimize the VAE's performance. In experimenting, we show how the VAE has learned the data distribution of the light curves by generating simulated light curves and its interpretability by visualizing attention scores and by visualizing the way the light curves are distributed along the continuous latent space using PCA. We show it orders the light curves across a smooth gradient from those those that have both low amplitude short-term variation and high amplitude long-term variation, to those with little variability, to those with both short-term and long-term high-amplitude variation in the condensed space. We also use PCA to display a potential filtering algorithm that enables parsing through large datasets in an intuitive way and present some of the pitfalls of algorithmic bias in anomaly detection. Finally, we fine-tuned the structurally correct but imprecise multivariate interpolations output by HeTVAE to three objects to show how they could improve constraints on time-delay estimates in the context of reverberation mapping for the relatively poor-cadenced ZTF data. In short, HeTVAE's use cases are ranged and it is a step in the right direction as far as being able to help organize and process the millions of AGN light curves incoming from Vera C. Rubin Observatory's Legacy Survey of Space and Time in their full 6 optical broadband filter multivariate form.

Declaration

The work in this thesis is based on research carried out at the Department of Physics, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

This work has used Durham University's NCC cluster. NCC has been purchased through Durham University's strategic investment funds, and is installed and maintained by the Department of Computer Science.
Based on observations obtained with the Samuel Oschin Telescope 48-inch and the 60-inch Telescope at the Palomar Observatory as part of the Zwicky Transient Facility project. ZTF is supported by the National Science Foundation under Grants No. AST-1440341 and AST-2034437 and a collaboration including current partners Caltech, IPAC, the Weizmann Institute for Science, the Oskar Klein Center at Stockholm University, the University of Maryland, Deutsches Elektronen-Synchrotron and Humboldt University, the TANGO Consortium of Taiwan, the University of Wisconsin at Milwaukee, Trinity College Dublin, Lawrence Livermore National Laboratories, IN2P3, University of Warwick, Ruhr University Bochum, Northwestern University and former partners the University of Washington, Los Alamos National Laboratories, and Lawrence Berkeley National Laboratories. Operations are conducted by COO, IPAC, and UW.

Contents

List of Figures

List of Tables

Dedication

Dedicated to my patient and magnanimous advisor Dr Hermine Landt-Wilman.

CHAPTER 1

Modeling Quasar Variability

1.1 Introduction

It is now understood that massive galaxies in our local universe are host to supermassive black holes (SMBH), which mainly grow through mass accretion, but clear details of this accreting region remain to be seen. A namesake of accreting black holes, active galactic nuclei (AGN), portrays their important role in the evolution of the galaxies that host them. Consider briefly that it is often the case that a rest-mass energy conversion of the inflowing matter of AGN outweighs the binding energy of the host galaxy (by an order of magnitude). Thus, if some marginal fraction of this energy is redispersed from the accretion process, there will be consequences for the host galaxy. The crux in our understanding of these regions lies in our inability to spatially resolve them, ironically despite AGN being the brightest known objects. This inner region typically resolves to sub-microacrsecond scales, surpassing the resolution capabilities of modern telescopes. One option has been X-ray interferometry studies, which played a role in imaging the BH shadow of M87. The capacity of X-ray interferometry likely will extend to the nearest Seyfert galaxies in the future, but will always elude the vast majority of AGN. As such, we consider an alternative, indirect approach called reverberation mapping.

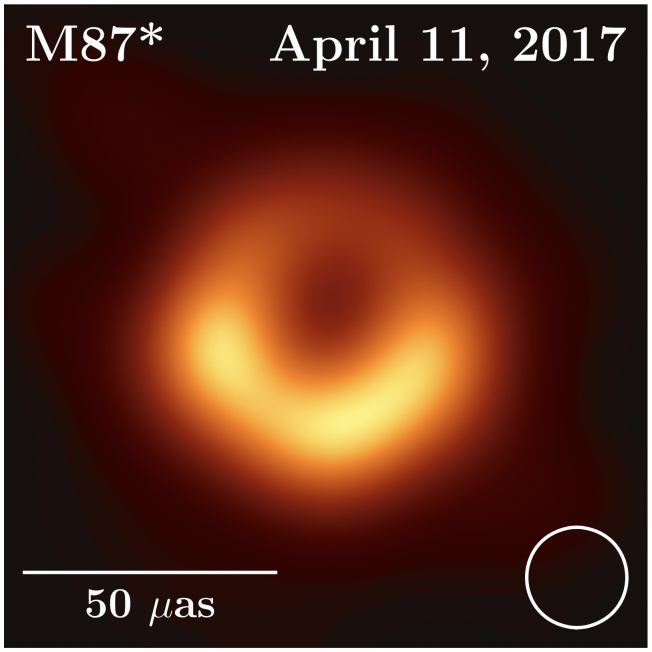


Figure 1.1: Depiction of the black hole shadow in M87 from the Event Horizon Telescope Collaboration through X-ray interferometry techniques (in units of brightness temperature).

The unleashing of gravitational potential energy through accretion causes X-ray and Extreme UV emission that originates close to the BH and is reprocessed as a thermal continuum in the accretion disc and as a broad-line emitting gas in a region sensibly named the broad line region (BLR). Reverberation mapping exploits that this energy release is both time-variable and that light travel time is constant to deduce structural information from the region by calculating a time delay from the observed emission line flux's response to the continuum flux's influence. The time delay (τ) is proportional to the size of the region so we can effectively swap spacial resolution for temporal resolution if we acknowledge that $R_{BLR} = c\tau$. Results from this have allowed us to deduce that the BLR is relatively small, as time delays from the BLR of typical Seyfert 1 galaxies are on the order of days to weeks.

The characteristic broadness of its emission lines is due to Doppler broadening as particles in this region are scattered, moving on the order of several thousand kilometers per second from being gravitationally bounded to the BH. If we assume Keplarian motion,

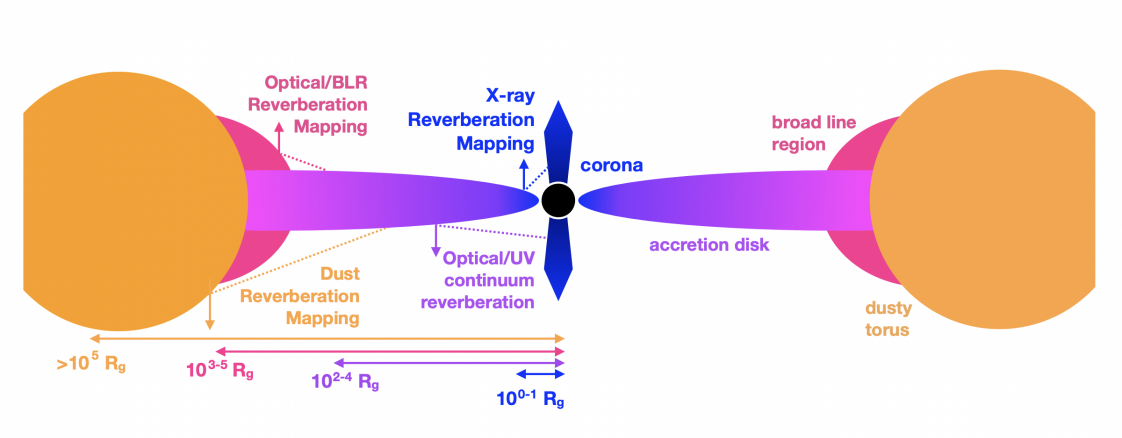


Figure 1.2: From [2], Visual with relative size scales for X-ray reverberation, optical/UV continuum reverberation, broad line region reverberation and dust reverberation.

the velocity of the gas can be determined with the line width as $v_{BLR} = f\Delta V$, where f is the order of unity depending on the overall geometrical distribution of the BLR clouds, their kinematics, and their line of sight emission properties. We can then use the estimated radius of the BLR to gather virial BH mass, $M_{BH} = \frac{R_{BLR}v_{BLR}^2}{G}$.

1.2 Quantifying Quasar Variability

If in reverberation mapping we aim to find correlations between associated variabilities of the continuum and BLR emission, we are heavily reliant on the 'amount' of variability in the observed sources, and albeit glossed over, the perceived variability itself is no small source of information about the inner details of AGN. The light curve data is a obviously a projection of the activity in AGN so understanding the data generating process behind it means understanding more about the underlying processes they are resultant of. A variety of models have been introduced to 'harness' it, some of which fortuitously provide an interpolation of the light curves that improves cross correlation techniques and estimating time delays. One key challenge facing the models are the aperiodicity of the variability signature as characterized by rapid, large amplitude stochastic flux variations on timescales from hours to years across the entire spectrum. What is more, the irregular sequencing of the light curves prompts technical difficulties because most sequential modeling procedures rely heavily on the assumption of uniform measurement cadence;

for instance those seen in Natural Language Processing tasks.

So, much less a model, estimating a magnitude value for the variability is a good way to pick out light curves that show meek variability or excessive variability. As a basic way to do this, we could take the ratio of the maximum flux and the minimum flux for a given light curve (R_{max}; [7]). The next step would be to compute a sample variance [8]:

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2 \tag{1.1}$$

Yet this metric does not consider additional variance from the measurement uncertainties. For that we remove the contribution from the errors to get the *excess variance* and *normalized excess variance*:

$$\sigma_{XS}^2 = S^2 - \overline{\sigma_{err}^2} \tag{1.2}$$

$$\sigma_{NXS}^2 = \frac{\sigma_{XS}^2}{\overline{x}^2} \tag{1.3}$$

The fractional root mean square variability (F_{var}) is also frequently used, but is a proxy of σ_{NXS}^2 and expresses the same information. Regardless, both are helpful as they convey an estimate of variability for a light curve beyond the contribution from measurement errors:

$$F_{var} = \sqrt{\frac{S^2 - \overline{\sigma_{err}^2}}{\overline{x}^2}} \tag{1.4}$$

Beyond this, finding a more complex model to pick out different aperiodic variability signatures would be helpful. Having a model means that, in a similar vein to having a magnitude of the variability, we can take a look at how the light curves are distributed with respect to the model parameters to find unusual sources or to see what types of variability is reflected by the majorities. We can discern distinct 'classes' of AGN that are representative of particular variability characteristics, and then compare new objects to these existing classes to see if their activity diverges from what has generally been seen

already.

Recently, Changing-state AGN (CSAGN) present a new challenge to catch anomalous variability in real-time. In this case, an AGN is changing over months to years from any one of several states or classification types to another: type 1 (showing broad permitted emission lines); type 2 (lacking them); objects that show large flux changes in their broad emission lines. As such, finding models that can detect the precursors to the changing states or forecast the transition is of interest to be able to direct attention to potential CSAGN candidates. Potential CSAGN candidates are chosen more or less through visual inspection currently, having unusual flaring activity or excessive change in optical flux. Other unobvious precursors could be unveiled with better models.

Also very often researchers will use the model to see how the variability changes (i.e. how the model parameters change) with respect to different physical properties of the black hole. Especially if the model parameter reflects some relatively intuitive aspect of the variability, then we can see directly how the variability changes from different black hole parameters. Ultimately, if robust enough correlations between model parameters and black hole physical parameters are found, then one can use the model fits to estimate black hole parameters with merely the light curves.

To mention anomalous activity again, if it is indeed shown that AGN variability is in large part a product of black hole physical parameters, then anomalous variability will only reflect anomalous black hole parameters, which is not necessarily a problem but distracts from finding more truthfully anomalous variability, like in CSAGNs. Balancing datasets by their black hole physical parameters is a clever way around this.

1.2.1 Carma Modeling

As of late, Continuous Auto-Regresssive Moving Average (Carma) models have been used to model and harness AGN variability. The variability can be thought of as abstracted by the model parameters so we can think about how these light curves are distributed

relative to this characterization to find interestingly variable sources. In addition, many any experiments have been performed to see how these models parameters, i.e. descriptors of variability, correlate with physical parameters of the black hole. This is a case where the model obligingly providing interpolations of the light curves to help with reverberation mapping studies by way of Gaussian processes.

The Carma(1,0) process specifically or Ornstein–Uhlenbeck process, or in astronomical literature, the Damped Random Walk model (DRW; [9]) has been the foremost used method to model AGN light curves. Being the most simple case of these stochastically volatile systems, the DRW is usually described by two parameters, τ, referred to as the relaxation time (or characteristic timescale; damping timescale; signal decorrelation timescale) and the amplitude of variability, σ. It entails a fixed-slope power spectrum density (PSD) on short timescales and a flat PSD at timescales beyond τ and its covariance function is $S_{nm}(\Delta t_{nm}) = \sigma^2 e^{-|\frac{\Delta t_{nm}}{\tau}|}$. It has also been described as the 'married drunkard's walk' in comparison to the 'drunkard's walk' because this particular drunkard (the DRW) has a tendency to 'come home' to spouse instead of drifting away: its random walk asymptotically shifts back to its mean [10].

Lately the suitability of the DRW model for AGN light curves has been called into question. For instance, [11] noted that fits for τ from many surveys have been biased in that the light curves need to be generally ten times the length of τ in order to properly constrain it. In addition, PSDs steeper than DRW PSDs have been shown on the high frequency timescales of a few months in Kepler light curves ([12], [13]) as well as in larger Pan-STARRS [14]. Discrepancies with the DRW PSD on low frequency timescales have also been found [15]. To instantiate a physical intuition, if either observed AGN variability is a projected result of more than one underlying process or multiple realizations of one, then the DRW model is will show discrepancies with AGN light curves [16]. Such results have led researchers to believe that the DRW model is too simplistic, inspiring several to investigate higher-order Carma models, whose PSDs are more aligned with AGN light curves. Namely, the Damped Harmonic Oscillator (DHO) [17], [18], [19].

Interestingly a DHO's equation of motion or physical inspiration is described by a spring with a drag force and displacement force both acting in the opposite direction of the displacement. The parameterization we will describe here for it is from [19], having four parameters with ξ as the damping ratio, ω_0 as the natural oscillation frequency, σ_ϵ as the amplitude of the short term perturbing white noise, and $\tau_{perturb}$ as the characteristic timescale of the perturbation process beyond which the perturbation process loses power. $\tau_{perturb}$ and σ_ϵ are conveniently kept to be equivalent to the associated DRW parameters.

As previously mentioned, Carma models are frequently used to help us relate black hole physical parameters with the observed AGN variability. Particularly, inverse relationships between optical variability and Eddington ratio, luminosity, and rest-frame wavelength have been found, but if these correlations have been found using the DRW model, then they are potentially spurious, either because the fits of the DRW are incorrect or because it is an inappropriate model for describing the variability. In the analysis of [19], more recent correlations have been been posed with DHO parameters, such as σ_{DHO} exhibiting an inverse relationship with Eddington ratio and black hole mass, and $\tau_{perturb}$ an inverse relationship with bolometric luminosity.

1.2.1.1 Limitations

Here we will briefly outline why some of the limitations of Carma modeling in a brief similar vein to [20]; if not for the sake of perspective of using these models to describe AGN variability, then at least for the sake of motivating the use of the deep learning methods described later on. For one, they question how models defined by Linear Stochastic Differential Equations (LSDEs) are insufficient to describe AGN variability that is for the most part dictated by nonlinear differential equations. Straightforward enough, but intuitively this means that there is a very limited range of functions that the models can describe, set by unphysically inspired assumptions we make of the variability. It would be better to have a physically inspired model that produces data, which we can then compare to perceived data. On a very high level we assume that the information garnered from

AGN variability can be compressed into the pair (or quartet) of parameters.

This stands in stark contrast to deep learning (DL) approaches that in a mathematically provable way can fit any functional form. And not that DL models have physical inspiration, but their unbiasedness can let the data attest to itself and more honestly compress things. In the unsupervised approach we will take later on, the worst assumptions we make are (1) that past observations help to guide future ones and (2) that variability can be expressed in a lower dimensional space, the size of which we can dictate.

1.3 Existing Reverberation Mapping Algorithms

The practiced methods we use to calculate time delays make a couple of key assumptions about the relationship between the continuum and BLR emission. Namely, that the continuum originates from a central source, that there is a relatively simple relationship between their observed fluxes and that the BLR cloud response time is instantaneous. These assumptions imply that we can quantify the relationship between a change in the continuum flux and the associated latent change in emission line flux by equation 1.5 [21].

$$\Delta L(t, V) = \int_0^\infty \Psi(\tau, V)\Delta C(t - \tau)d\tau \tag{1.5}$$

At a particular time delay and line-of-sight velocity, Ψ measures the response that a given change in the continuum flux would necessitate for the emission line flux. But in practice, we usually settle for the mean time scale response by cross-correlating the continuum and emission-line light curves as a means to estimate the centroid of the delay map. In the case of spectral light curves, this means maximizing the following function.

$$CCF_{XY}(\tau) = \frac{Cov(X_t, Y_{t-\tau})}{\sqrt{Var(X_t)Var(Y_{t-\tau})}} \tag{1.6}$$

A similar analysis can be done with photometric light curves, except that the broadband filters cover both the continuum and emission lines. Now instead we would search for the

maximum of the difference between the CCF of the two light curves and the ACF of the continuum light curve [22], where the ACF is the CCF between a light curve and itself, describing how the light curve is related to itself at its past observations or lags.

$$CCF(\tau) = CCF_{XY}(\tau) - ACF_X(\tau) \tag{1.7}$$

Unfortunately what hinders the efficacy of this cross correlation analysis is that it requires data to be uniformly sequenced, but ground based telescopes are quite restricted by dinural and seasonal restraints and weather permissions, often resulting in very sparse and irregular measurements across time. To mitigate this, the interpolation cross-correlation function (ICCF; [23]) was introduced, which linearly interpolates the driving continuum light curve before measuring time delays. This method falters in so much as it depends heavily on the sampling quality of the driving light curve, but it still very frequently used to estimate reverberation lags. Also earlier on, the discrete correlation function (DCF; [24]) was proposed, which calculates the correlation at all of the discrete time points available and places the values of the correlation at different lags into equally spaced bins.

1.3.1 ZDCF

The Z-Transformed Discrete Correlation Function (ZDCF; [25]) is an improvement on the DCF's faulterings such that it adapts the bin sizes relative to the number of observations in the light curve so that not all bins are equal in time-lag width, provides robust error estimates and performs a Fisher's z-transform to improve estimates for the CCF. The Fortran implementation [26] of the ZDCF is offered as well as code to detect the maximum likelihood peak location (PLIKE).

1.3.2 Javelin

JAVELIN, a python implementation of SPEAR [27], models the variable source's driving continuum light curve using a Damped Random Walk (DRW). The algorithm takes a

different approach than typical cross-correlation in that it applies a top hat filter to the DRW fit of the driving light curve, then scales the light curve across its magnitude and shifts it by a time-delay to match the emission-line light curves. Consequent parameter estimates for the top-hat width, scaling factor and time-delay are made using Markov Chain Monte Carlo algorithms (MCMC). Its worth noting that it takes the same approach as the ICCF does in that it is particularly dependent on the continuum light curve to guide the lag estimation.

The authors acknowledge that the relationship between driving and emission-line light curves can be nonlinear, but that 'the amplitude of variation on reverberation timescales is sufficiently small that the linear approximation seems to be justified' [27]. Relative to CCF methods, Javelin yields tighter uncertainties in the time delays [28], potentially due to a more appropriate model of the driving light curve. Javelin adds no particular scaling to the error bars, assuming they are already accurate; a case which is considered subsequently in PyROA.

1.3.3 PyROA

PyROA [29] introduces a running optimal average (ROA) in order to model light curve data with a holistic approach, considering data from each associated light curve via 'stacking' to form the general model and subsequently derive lags, making it less dependent on the sampling quality of the driving light curve. Compared to Javelin by way of a DRW, no assumptions are made about the shape of the driving light curve; the shape is modeled and smoothed explicitly using the data provided.

The running average is an inverse-variance weighted (W_i) average of the N observations described by the tuple t_i, D_i and errors σ_i shown in 1.8a. 1.8b reflects the Gaussian window function that the model includes to *blur* light curves, reducing the relevance of points relative to their temporal distance and error values according to the window's width, Δ. Equation 1.8c shows the variance for the model. With a singular 'stacked' model, $X(t)$, the window is the same across all the light curves.

$$X(t) = \frac{\sum_{i=1}^{N} D_i W_i(t)}{\sum_{i=1}^{N} W_i(t)} \tag{1.8a}$$

$$W_i(t) = \frac{1}{\sigma_i^2} e^{-\frac{1}{2}\left(\frac{t-t_i}{\Delta}\right)^2} \tag{1.8b}$$

$$Var(X(t)) = \frac{1}{\sum_{i=1}^{N} W_i(t)} \tag{1.8c}$$

A smaller value of Δ in effect implies a tighter fit, while a larger one, a less precise fit. Similarly, smaller value of Δ implies more effective model parameters, while a larger one, less. The balancing act imposed by this parameter between underfitting and overfitting is consequently decided by minimizing the Bayesian Information Criterion statistic (BIC). Individual Light curve fits are ascertained for each light curve by shifting the ROA by a given τ, scaling by the relevant rms A and shifting by the relevant mean B 1.9. For such parameters, the *emcee* [30] MCMC sampler is used with uniform priors and the BIC as its likelihood.

$$f_i(t) = A_i X(t - \tau_i) + B_i \tag{1.9}$$

With the same blurring knob across all the light curves controlled by Δ, the transfer function is by default a Dirac Delta. But, going beyond this, PyROA allows for other transfer functions to be used, interestingly like those of: Gaussian, log-Gaussian and uniform, which allow for more malleability and blurring that is optimal for each light curve individually, and in effect, introduces the parameters, τ_i, the mean lag and Δ_i, rms of the delay distribution. PyROA additionally accounts for potentially underestimated observational measurements of the light curves so that error parameters are adjusted with an extra term, in this case, s (equation 1.10). The ROA's noise model handles outliers via sigma clipping. This sets the chi-squared values of data points outside a reasonable threshold ($N\sigma$; 3σ lies outside a probability of 99.7%) to be constant, which equivalently expands the datum's error bars to be exactly the same as the sigma threshold.

$$\sigma_{j,i} \rightarrow \sqrt{\sigma_{j,i}^2 + s_i^2} \tag{1.10}$$

CHAPTER 2

Deep Learning

DL has emerged as a general framework for learning complex nonlinear functions. The main advantages of DL models are that they exhibit a much weaker bias than Carma models, injecting no information in order to fit the data. The staple multilayer perceptron (MLP) or standard neural network (NN) is able to map any number of input features to an output value using weights in hidden layers and nonlinear *activation* functions. The weights are then optimized for an objective function or cost function via some form of gradient descent. Different adaptations of NNs like convolutional neural networks (CNNs) work through the same fundamental principles, but improve upon MLPs for particular datasets because they exhibit a better inductive bias to capture the relationships in the data, which serves to steer the network in the right direction to learning the desired functional mapping. One result of a CNN's inductive biases is translational equivariance, which means that if the input is translated, the output will reflect the translation. In images, particularly in object detection, this is desirable because if the object moves, the output should detect the movement, but still acknowledge the object. If an image is input into an MLP, it is flattened first. If this same image is translated in some way and then flattened, the input is totally out of order, and the MLP will have to go quite out of the way to accommodate for this. The CNN maintains its translational equivariance

by applying the same 'filter' of weights across the input image. This implies that local relationships are captured relative to the size of the filter and that the outputs do not change with respect to location.

Sequential models assume that the input data is instead sequentially related to provide an advantage over MLPs. The evolution from the subsequently mentioned Long short-term memory (LSTM; [31]) networks to the Transformer [3] can be condensed into injecting a better inductive bias in our models because LSTMs struggle to capture long term dependencies, while Transformers make it less difficult to account for any timescale dependency. Significant amounts of money and research have gone into sequence-to-sequence models that assume regularly spaced-input because of the utility for applying these models to language problems like neural translation. Once the principles of modern sequential models for deep learning are established, the nontrivial adaptations they must make for irregular time series can be discussed.

2.1 From Recurrence to Attention (solely)

Prior to the introduction of the Transformer, sequential models in deep learning relied on recurrent architectures that persisted relevant information from previous data points to learn temporal dependencies. Each position of the sequence is processed one by one, generating a hidden state $h^{<t>}$ relative to previous hidden state $h^{<t-1>}$ and the input $x^{<t>}$. The output for the cell is the hidden state passed through a linear layer and a softmax to generate probabilities for the prediction $y^{<t>}$. In the case of modeling natural language, the index of this vector with the highest probability corresponds to the word that the network thinks is most likely next.

$$\begin{aligned} h^{<t>} &= tanh(W_h[h^{<t-1>} | \, x^{<t>}] + b_h) \\ o^{<t>} &= W_o h^{<t>} + b_o \\ y^{<t>} &= softmax(o^{<t>}) \end{aligned} \tag{2.1}$$

Improvements to 'vanilla' recurrent neural networks (RNNs) sought to address the prob-

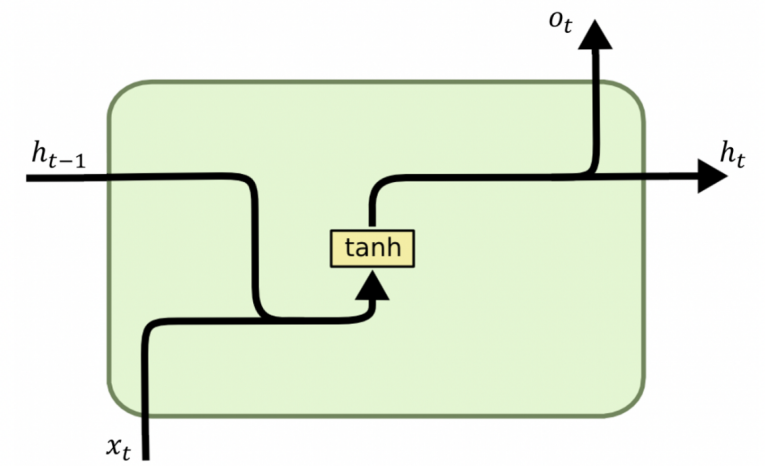


Figure 2.1: Depiction of 'vanilla' recurrent cell and corresponding equations.

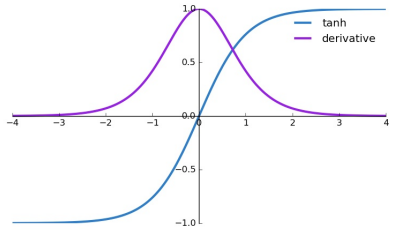


Figure 2.2: Depiction of tanh and its associated gradient. Note the saturating regions in the gradient graph on both tails.

lem of vanishing gradients, which hampered the networks ability to learn long-term dependencies. As the error is propagated backwards in the network, the earlier gradients can become exponentially small due to the chain rule in conjunction with the gradient structure of applied nonlinear functions like *sigmoid* and *tanh*. For instance, the *tanh* function has gradient values in the range (0,1] with large saturating regions on its boundaries so the gradients are almost zero in these regions. As backpropagation through time progresses, these small numbers are multiplied, resulting in insignificant weight updates for the earlier layers and thus dysfunctionally slow learning. While in theory a recurrent architecture could carry relevant information over from an unfixed latency, plain RNNs do not pose a better solution than archetypal MLPs with limited time windows because of this vanishing gradients problem.

Consequently, LSTMs were introduced and are able to pass along relevant information

$$
\begin{aligned}
\widetilde{C}^{<t>} &= \tanh(W_c[h^{<t-1>}, x^{<t>}] + b_c) \\
i_t &= \sigma(W_i[h^{<t-1>}, x^{<t>}] + b_u) \\
f_t &= \sigma(W_f[h^{<t-1>}, x^{<t>}] + b_f) \\
o_t &= \sigma(W_o[h^{<t-1>}, x^{<t>}] + b_o) \\
C^{<t>} &= i_t * \widetilde{C}^{<t>} + f_t * C^{<t-1>} \\
h^{<t>} &= o_t * \tanh(C^{<t>}) \\
y^{<t>} &= \text{softmax}(W_y h^{<t>} + b_y)
\end{aligned}
\tag{2.2}
$$

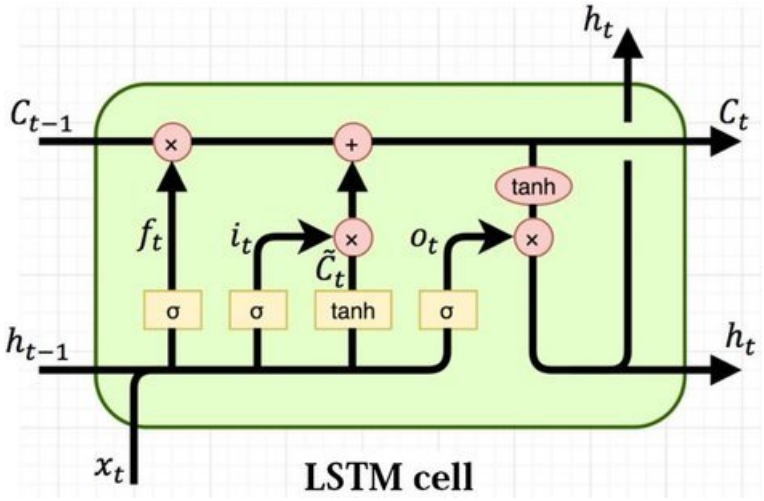


Figure 2.3: Equations and associated image for forward propagation in an LSTM cell.

over longer input sequences to handle 'long-lag' problems using memory 'gates,' while leaving the recurrent, short-term feedback loop structure unchanged; hence the naming convention: *Long short-term Memory* network. The gates allow the network to more intuitively weight or control the influence of previous input and the current input regarding what information it would pass along from current input, what it would 'forget' from previous input and what of this mixture it would output, helping to reconcile the vanishing gradients problem in addition to *weight update conflicts* in standard RNN architectures. Intuitively, in any instance where a long lag dependency is trying to be established by updating the weights that have already learned short-term structure of the sequence, there will be a weight update conflict. More gates mean means less weight update conflicts and more control over the flow of information in the network.

The LSTM's true potential has not been received until relatively recently. Antecedent to the first GPU (1999), it remains relevant almost 20 years later. Although the Gated Recurrent Unit (GRU; [32]) was introduced and reduced the number of cell 'gates' to two, the LSTM still outperforms it on many tasks. In the case where the two different architectures perform similarly, it is worthwhile to instead use the GRU because it trains faster as a result of having a relatively smaller parameter count.

Many modern add-ons to traditional neural network architectures also contribute non-detrimentally towards the vanishing gradient problem. For instance, batch normalization takes each output layer's values before a typical nonlinear function is applied and normalizes them to fit comfortably inside the saturating regions, where these values have more substantial gradients to update the network with. Moreover, the Rectified Linear Unit (ReLU; [33]) activation function has been introduced. It has values of zero when $x < 0$ and a line with a slope of one at x values greater than zero. Its constant slope means stepping down the gradient to update model weights is not as impeded and learning is not diminished by these saturating regions. Weight initialization has also proven itself to be at the crux of training NNs. The weight initialization used by Hochreiter and Schmidhuber (sampling $[-.2, .2]$ or $[.1, .1]$) was impressively not far off from today's ideologies like Xavier initialization [34], which for a *tanh* activation means sampling $[-1/\sqrt{N}, 1/\sqrt{N}]$ for each weight with N as the number of input nodes to the layer.

The next several paradigm shifts for sequence models in deep learning have been for the sake of Seq2seq problems like neural machine translation. To handle these problems an input sentence is passed through an RNN word by word, and the last output vector, say $h^{<J>}$, is considered as a summary of this sentence, which is used by another RNN to generate the translated sentence. This system of using a compressed summary of the input sequence to produce an output sequence is referred to as an encoder-decoder style architecture.

Recognizing this single output vector as a bottleneck to learning, Attention mechanisms were introduced [35]. Interestingly, the now ubiquitous term 'attention' was not established until some time after the fact. The thought process behind it was that the network should not only construct a better summary to pass to the decoding layers, but also those decoding layers should have better access to said summary. The network should be able to refer back (or pay attention) to what was relevant in the original sentence at its leisure.

But before 'attention' stood on its own, it was used in cahoots with recurrent architectures. In such Attention-RNNs [35], the attention mechanism happens in between the encoder and decoder layers. The idea is to create context vectors, $c^{<i>}$ (not to be confused with the LSTM cell state C), that are passed as input for each decoder block (1 to I) and contain the most relevant information for that specific decoding step from the input sequence.

To generate $c^{<i>}$ for one of the decoding layers: all output vectors $h^{<j>}$ from the encoding steps (1 to J) are concatenated with the previous hidden state of the current decoder block, $s^{<i-1>}$, as$[s^{<i-1>}|\ h^{<1>}], [s^{<i-1>}|\ h^{<2>}]$ ...$[s^{<i-1>}|\ h^{<J>}]$.

Each of these concatenated vectors are passed through a linear layer and normalized with a softmax, capturing the degree of relevance every output from the encoding phase, $h^{<j>}$, has to the input $s^{<i-1>}$ in the form of a probability score, $\alpha_{i1}, \alpha_{i2}, ...\alpha_{iJ}$.

(Linear layer)

$$e_{ij} = W[s^{<i-1>}|\ h^{<j>}] + b \tag{2.3}$$

(Softmax)

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{j=1}^{J} exp(e_{ij})} \tag{2.4}$$

Each $h^{<j>}$ is then scaled by its relevance score, and the scaled vectors are added together to create $c^{<i>}$.

$$c^{<i>} = \sum_{j=1}^{J} \alpha_{ij} h^{<j>} \tag{2.5}$$

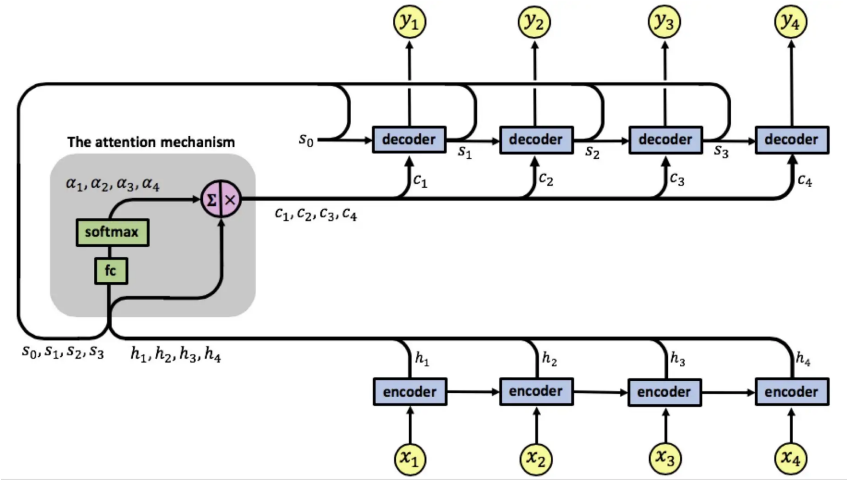


Figure 2.4: Depiction of an Attention-based RNN. Consider that we are interested in calculating $c^{<1>}$. We would take the concatenated vectors $[s^{<0>}|\ h^{<1>}], [s^{<0>}|\ h^{<2>}], [s^{<0>}|\ h^{<3>}],\ [s^{<0>}|\ h^{<4>}]$ and pass them through a linear layer, creating $e_{11}, e_{12}, e_{13}, e_{14}$. Softmaxing these values gives us $\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}$, to get $c^{<1>} = h^{<1>}\alpha_{11} + h^{<2>}\alpha_{12} + h^{<3>}\alpha_{13} + h^{<4>}\alpha_{14}$ and voila.

Lingering still is that in recurrent models, each input of a sequence has to be processed serially or one at a time, precluding intra-example parallelization. If parallelization within examples is not possible, our ability to perform mini-batching is reduced as sequence lengths get larger and larger because such examples have to exist in the same finite-sized memory. Concerns like these over the RNN's speed and scalability sparked a need for parallelized sequence models in which inputs could be perceived and dealt with simultaneously. Google brain introduced the Transformer in 'Attention is all you need' introduced and novel in so much as it disposed of recurrence entirely, relying solely on attention (with some bells and whistles).

In attention-based recurrent networks, we calculate attention between the previous hidden state in the decoder and each output of the encoder to get a compressed summary to input into that decoding step. In the Transformer, attention is calculated betwixt the input sequence itself to create a compressed summary that can be 'queried' at each decoding step without recurrence (at least in the encoder). Moreover, instead of concatenating

the vectors, passing them through a linear layer and softmaxing them to get attention scores, attention is calculated by passing the individual vectors through a linear layer (actually two with a ReLU in between), computing a scaled dot product between them and then softmaxing them. The advantage that the dot product attention has is that it is faster and more memory efficient such that it can be implemented with matrix multiplication, wrapping up the calculations neatly as $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$, with Q,K,V all being the post-linear layer representations of the input sequence. In the case of translation and NLP tasks, ML-friendly representations of the input data are word embeddings, and the linear layers can scale them to whatever dimensionality is preferred before 'self-attention'. Q,K,V are then each N (the length of input sequence) by d (the length chosen to scale input representations to).

The Transformer network's 'Scaled Dot-Product Attention' imposes a scaling factor $\sqrt{d_k}$ to prevent the vanishing gradient problem if the matrix multiplication of Q and K returns large dot products. Analogous to databases, the attention function maps query values to key values, intuiting a question and answer that each step in the sequence can ask and respond to the other. The output is a weighted sum of the values (V) which contains relevant information of adjacent dependencies in a sequence at each step in the input sequence. The Transformer use of 'self-attention' to connect or relate all input positions requires a constant number of operations compared to recurrent layers that require $O(n)$ sequential operations. This lets one consider the entire input sequence simultaneously, while decreasing the path length that forward and backwards signals have to traverse find correlations between different positions in an input sequence. Prior attempts to reduce sequential computation employed convolutions to garner representations in parallel and were successful in doing so, limiting the number of operations required to find correlations from input and output positions as the distance between them grew: Linearly for ConvS2S, and Logarithmically for ByteNet. The Transformer took a leap forward in reducing this computation to constant time.

For the incident sequence in the previous figure, $x^{<1>}$ to $x^{<4>}$, each position is trans-

Figure 2.5: Self-attention visualization from [3], opacity shows relevance proportions. It is important to note as well the interpretability of attention distributions that show clear grammatical and semantical structure, not a quasi-illegible latent vector like an LSTM encoder would output, especially as the 'black-box' nature of many deep learning algorithms can be unsettling or unacceptable in many domains.

formed by the shallow MLPs (Q,K,V each have their own) to get $q^{<1>}$ to $q^{<4>}$, $k^{<1>}$ to $k^{<4>}$, $v^{<1>}$ to $v^{<4>}$ all of a chosen dimension d. For each q, scaled dot products are taken with $k^{<1>}$ to $k^{<4>}$, as $e_{11} = q^{<1>}k^{<1>}$,..., $e_{14} = q^{<1>}k^{<4>}$ and scaled by $\sqrt{d}$ before being softmaxed to $\alpha_{11}, ..., \alpha_{14}$. The probability scores α, or relevance each k has to a particular q are scaled by the values and combined to get $\text{attention}(q^{<1>}) = v^{<1>}(\alpha_{11}) + v^{<2>}(\alpha_{12}) + v^{<3>}(\alpha_{13}) + v^{<4>}(\alpha_{14})$.

Thus the resultant attention embedding is a matrix $\textbf{attention}(Q, K, V)$, whereby each column refers to the attention scores of dimension d for a particular step in the input sequence.

The next step is stacking multiple attention blocks together to form **Multi-head Attention**. Each attention head is a different linear projection of the original input sequence representation and conjoined as: $\textbf{Multi-head}(Q, K, V) = Concat(head_1, ..., head_h)W^O$ with $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$. The intuition here is that different questions, Q, pose different answers, K, that the network should pay attention to relative to the input; it 'allows the model to jointly attend to information from different representation subspaces at different positions' [3] or different layers of conceptual abstraction. One word might provide the most relevance to another, or the best answer in one way, and another word might provide the best answer to the original word with a different question. In the decoder block, the keys and values are passed from the encoding block of the input sentence, while the queries come from what the decoder has output so far. In training we have the entire target sequence, but to simulate a test case where we would not, the queries for values not predicted yet are masked. This process begins with a $< sos >$ (start of sentence) token.

Transformer-esque models 'inject' time into input embeddings because learned self-attention does not contain order information (the representation could be shuffled without loss of information). As such, each relative position PE_{pos+k} is linear function of PE_{pos}. These positional encodings can be learned or fixed, and the authors choose a fixed sinusoidal version in this model as their performance was the same.

$$PE_{pos} = \left\{ \begin{array}{ll} sin(pos * \omega_i) & i \text{ is even} \\ cos(pos * \omega_i) & i \text{ is odd} \end{array} \right\} \tag{2.6}$$

$$\omega_i = \frac{1}{10000^{\frac{2i}{d_{model}}}} \tag{2.7}$$

To make predictions the representation of the input sequence is used as both the K and V values, and the Qs are the labels, or the translated sentence. Transformers are still

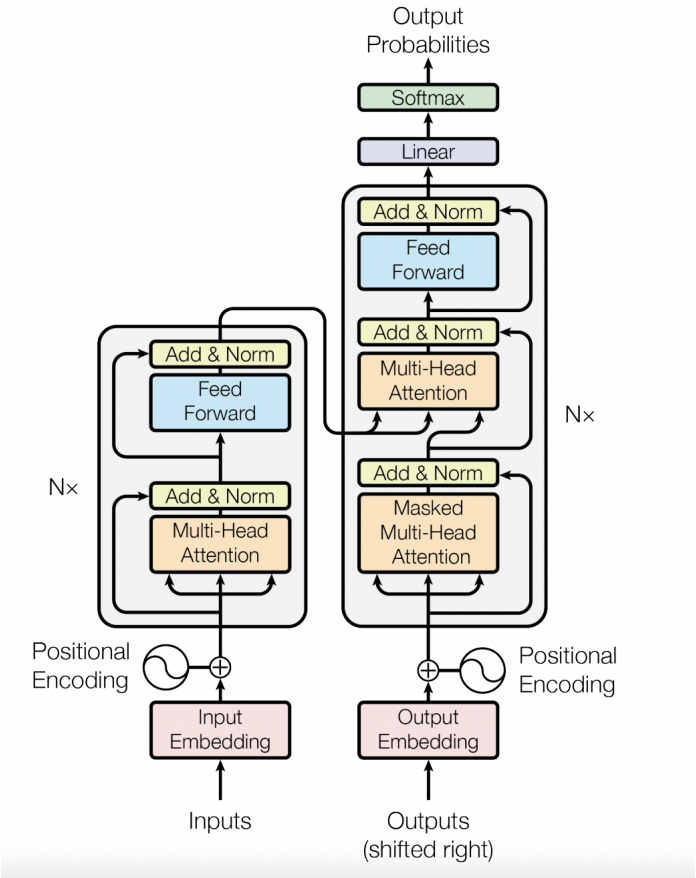


Figure 2.6: From [3], the Transformer architecture. A couple of extra things to note: all of the residual connections incorporated; the embedding feeding in on itself N times.

limited by giving output recurrently, whereby to ensure that each subsequent prediction is only dependent on the already translated words, future Qs are masked (setting values to $-\inf$). In the initial phase of prediction, all the Qs are masked except the $< sos >$ token, then the first word is predicted, all of the other corresponding Qs are masked and so on.

To reiterate, attention blocks provide the network with an ability to weight, or 'pay attention' to the most relevant parts of the input sequence for learning, irrespective of sequential or temporal 'distance.' Self-attention 'is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence' and it is the first architecture to use attention mechanisms solely. As far as the name 'Transformer' is concerned, it's meant to imply itself as the next paradigm of sequential

modeling. The field is henceforth 'transformed,' and the authors believe 'attention-only' architectures are the inevitable next step in deep learning advancements. Moreover, its naming is relative to the transductive tasks it can be applied to.

The only drawback lies in the rare case that the input representation dimension d is larger than the sequence length as the computational complexity per layer of self-attention is $O(n^2d)$ compared to recurrent networks $O(nd^2)$. But, if input sequences are incredibly long self-attention can be administered to a neighborhood size around each step in a sequence.

2.2 From AEs to VAEs

Deep learning has altered the field of machine learning to the extent that complex feature representations can be learned by the model without needing the expertise that would be required otherwise. Specifically, auto-encoder modules (AEs) are able to learn informative and compressed summaries of input data in the form of a discrete, finite length vector. To do this, input data is propagated through a sufficiently deep network with a 'bottleneck' layer, called an *encoder*, while a second NN, called a *decoder*, tries to reconstruct the original data from its compressed form. Sensibly, if the network is to minimize the reconstruction error, it must find the most compact representation of the data to go through the suffocating layer.

Apart from data compression, AE concepts are used ubiquitously in DL tasks such as image segmentation, image inpainting and data denoising in addition to the seq2seq procedures previously mentioned. For instance in data denoising, one would add noise to the input data and pass it through the autoencoder, comparing it with the denoised labels in order to teach the network to denoise the data on its own.

To instigate a motivation for Variational Autoencoders (VAEs; [36]), consider that you have a trained AE for your data, and as such, you have a series of latent space vectors for each example that are distributed in some way. With some investigation, you might

find that the vectors are very unintuitively spaced because the network has no constraint on this space and will do whatever 'cheating' it can to more easily reconstruct the data. If you chose a random vector to decode in the realm of this space you would be returned something likely very incongruent to the rest of the data.

Ideally, the distribution of the latent space should be both continuous and complete so that two vectors near each other decode similarly and every vector in the space decodes meaningfully. VAEs accomplish just this, building upon standard AEs by altering the encoder to output a multivariate Gaussian distribution (two vectors for μ and σ) instead of a discrete vector and changing the cost function to place a constraint on the latent space in order to permit random sampling and structured uncertainty. And while considering a VAE as an autoencoder with a constrained latent space is a simple enough conceptual way of thinking about it, at the heart of VAEs is variational inference, and understanding how the VAE creates a continuous and complete latent space necessitates understanding variational inference.

Variational inference is a technique used to approximate a complex distribution by choosing the most comparable option from a family of parameterized distributions that are reasonable to work with. The complex distribution we are interested in approximating is the compressed distribution of the data: a latent random variable z that we would like to infer from some observed data x. Let us first gander at the expanded form of Bayes rule:

$$\begin{aligned} p(x,z) &= p(x,z) \\ p(z|x)p(x) &= p(x|z)p(z) \\ p(z|x) &= \frac{p(x|z)p(z)}{p(x)} \end{aligned} \tag{2.8}$$

The conditional distribution, $p(z|x)$, or the posterior is desirable to compute as it represents the distribution over the latent variable conditioned on the observed data, i.e. *a posteriori*. The likelihood $p(x|z)$ is proportional to the posterior and is the probability

of the data given particular values for the latent variable. The prior $p(z)$ constitutes our inductive bias on the latent space's distribution. The problem lies in computing the normalizing constant, or marginal probability $p(x)$, which is oft intractable with high-dimensional, 'real-world' datasets not of simple functional forms or tractable conjugate cases because one must integrate over every possible value of the latent variable.

$$p(x) = \int_{z_0} ... \int_{z_D - 1} p(x|z)p(z)dz_0...dz_{D-1} \tag{2.9}$$

To circumvent this we can try to use a surrogate, $q(z)$ with a familiar Gaussian form to approximate the possibly complex, multi-model posterior distribution. Then it would be convenient to be able to compare our posed surrogate distribution $q(z)$ with $p(z|x)$. The Kullbeck-Leibler (KL) divergence does exactly this, measuring a non-negative distance between two given distributions, while being zero if the distributions are equal. Thus setting up the following optimization problem:

$$q^*(z) = \underset{q(z) \in Q}{arg\ min}(KL(q(z) \| p(z|x)) \tag{2.10}$$

From the family of distributions we chose, $q(z) \in Q$, we are looking for the optimal one $q^*(z)$ that minimizes the KL divergence from the posterior. Unfortunately we have created another problem: how can we compare the proposed distribution to the posterior if it is unknown? First we can look at the KL divergence between the posterior and the supposed $q(z)$ in equation 2.11.

$$
\begin{aligned}
KL(q(z)\|p(z|x)) &= \mathbb{E}_{z\sim q(z)}[log\frac{q(z)}{p(z|x)}] \\
&= \mathbb{E}_{z\sim q(z)}[log\frac{q(z)p(x)}{p(z|x)p(x)}] \\
&= \mathbb{E}_{z\sim q(z)}[log\frac{q(z)p(x)}{p(z,x)}] \\
&= \mathbb{E}_{z\sim q(z)}[log\frac{q(z)}{p(z,x)}] + \mathbb{E}_{z\sim q(z)}[log\, p(x)] \\
&= \mathbb{E}_{z\sim q(z)}[log\frac{q(z)}{p(z,x)}] + log\, p(x) \\
&= -\mathbb{E}_{z\sim q(z)}[log\frac{p(z,x)}{q(z)}] + log\, p(x) \\
&= -L(q) + log\, p(x)
\end{aligned}
\tag{2.11}
$$

After some rewriting, the quantity on the l.h.s. is dependent only on the distribution q so it can be simplified to just $L(q)$. It also is entirely computable considering that it is not dependent on the marginal, $p(x)$, or the posterior. The r.h.s. is the log of a probability distribution, which is a fixed value, as it does not change with our surrogate distribution, and will be negative, as a distribution only takes on values $[0, 1]$, and the log of a value in $[0, 1]$ is always negative. Thus, for the KL divergence to be positive, $L(q)$ has to be negative; in fact smaller than the evidence, log $p(x)$. Hence L(q)'s namesake, the Evidence Lower Bound (ELBO).

$$
\begin{aligned}
KL(q(z)\|p(z|x)) &= log\, p(x) - L(q) \\
&= \text{evidence} - ELBO
\end{aligned}
\tag{2.12}
$$

If the evidence is fixed, then maximizing the ELBO implies minimizing the KL divergence, or minimizing the KL is an equal optimization problem to maximizing ELBO. This is all well and good to see how variational inference works, but how does this relate to the neural network form of the variational autoencoder? How will we calculate the ELBO via a neural network architecture?

$$
\begin{aligned}
q^*(z) &= \underset{q(z)\in Q}{arg\ min}\, KL(q(z) \,\|\, p(z|x)) \\
&= \underset{q(z)\in Q}{arg\ max}\, L(q) \\
&= \underset{q(z)\in Q}{arg\ max}\ \mathbb{E}_{z\sim q(z)}[log\frac{p(z,x)}{q(z)}] \\
&= \underset{q(z)\in Q}{arg\ max}\ \mathbb{E}_{z\sim q(z)}[log\frac{p(x|z)p(z)}{q(z)}] \\
&= \underset{q(z)\in Q}{arg\ max}\mathbb{E}_{z\sim q(z)} log\, p(x|z) + \mathbb{E}_{z\sim q(z)} log\, p(z) - \mathbb{E}_{z\sim q(z)} log\, q(z) \\
&= \underset{q(z)\in Q}{arg\ max}\mathbb{E}_{z\sim q(z)} log\, p(x|z) + KL(q(z) \,\|\, p(z))
\end{aligned}
\tag{2.13}
$$

Revisiting the ELBO term in 2.13, we see inexplicitly approximating the posterior with $q(z)$ by maximizing the ELBO entails calculating the KL divergence between the prior distribution and the proposed distribution and calculating the likelihood of the data. In the framing of a VAE, the encoder is what computes the proposed $q(z)$, and the decoder is what computes the likelihood $p(x|z)$. The encoder is a neural network that outputs vectors for each μ and σ, which are used to form a multivariate Gaussian whose covariance matrix is a diagonal matrix with the diagonal entries given by the exponentiated output of the decoder network (for simplicity's sake). The decoder is another neural network that tries to reconstruct the original data from the latent space sample, and whatever it reconstructs, we can then determine how characteristic it is of the actual data. Equivalently, the decoder maximizes the expected log likelihood of x given z when it is sampled from $q^*(z)$ by passing z through a neural network that is trained to minimize the mean squared error between the predicted reconstruction of x and its known values.

The KL term from 2.13 can just be added to the neural network's cost function because we decide the prior over our latent variable z to follow a familiar functional form and have access to the current approximate of the posterior, $q(z)$. Specifically, the KL between the two Gaussian's, $q(z)$ and $p(z)$, is shown in equation 2.14.

$$KL(q(z)||p(z)) = log\frac{\sigma_{p(z)}}{\sigma_{q(z)}} + \frac{\sigma^2_{q(z)} + (\mu_{q(z)} - \mu_{p(z)})^2}{2\sigma^2_{p(z)}} - \frac{1}{2} \tag{2.14}$$

It is worth mentioning that sampling from the latent space during training using the obvious parameterization, of μ and σ, is non-differentiable. The solution to this is the *reparameterization trick*, which relies on an auxiliary noise term to compute the sample and thus connect the encoder and decoder neural networks.

$$z = \mu + \sigma\epsilon; \quad \epsilon \sim \mathcal{N}(0, I) \tag{2.15}$$

In the case of a Gaussian, the valid and backprop-able reparameterization is shown in equation 2.15.

So once the network is trained, we have a distribution $q(z)$ that approximates the posterior $p(z|x)$ over the latent space, thus ensuring it as a continuous distribution that we can sample from. Anywhere in this continual distribution is fair game with regard to being able to reconstruct a sample from it and getting something not unlike the dataset that the network was trained on. This is the generative aspect of VAEs.

2.2.1 Self-Supervision vs. Unsupervision

We have seen how attention differs from recurrence in that it is better able to accommodate a multitude of different time-scale relationships, and we have seen how a variational autoencoder intuitively constrains the latent space into a smooth distribution as compared to a standard autoencoder, but the novelty of training a neural network in a self-supervised way should also be discussed briefly.

Self-supervised learning can be thought of as under the umbrella of unsupervised learning in so much as it uses the structure of the data itself to learn, but its technique usually works by hiding a subsample of the data from the network and making predictions for

specifically that hidden data. To relate this to the masking procedure in [3], The Transformer's training task does not hide any information in the encoding scheme: it uses the full input data and masks known output word embeddings so that they are not used prematurely in the decoding steps, i.e. when predicting the second word in the output sequence, it uses the entire attention-encoding from the input sequence and the first word in the output sequence to make its prediction. The self-supervised version of this would be to hide part of the input sentence when creating the encoding, predicting to the words which are hidden.

The advantages of self-supervised learning are that the network learns more semantically relevant representations of the data and so also better performs on downstream tasks with it being standard now to use in large scale language models [37].

2.3 Existing Methods

Recurrent AEs have proven themselves to be successful in many tasks in astronomy. For example, they have been successful as feature abstractors for the sake of stellar classification using light curves from optical variable stars [38]. In the case of extracting insight from Quasar Variability, [39] has set the precedent and proved their efficacy over the DRW by training an LSTM RNN on approximately 15,000 light curves from the Catalina Real-time Transient Survey.

In this work, the authors first project the latent space representations of the light curves from the trained AE onto two dimensions using principal component analysis (PCA). They then placed a gridded surface on the 2-D space and averaged the latent vectors in each 'square.' This gave them an idea of the types of variability that were represented in each of the regions and what types of variability were reflected by the majority of the dataset versus the minorities. Following this, they used physical parameters corresponding to the light curves like black hole mass and optical luminosity and found the most correlated dimension of the latent space vectors with each parameter respectively. They did this by passing the latent vectors through a one layer MLP trained to maxi-

mize the coefficient of determination value, R^2 for each parameter. Then all but one of the latent vector's values were zeroed out and passed through the same MLP, and presumably whichever dimension of the latent vector returned the highest R^2 was the one most correlated with the associated physical parameter. Finally, by toggling the value for this dimension of the latent vector and reconstructing the light curve with the decoder, they could see by proxy how the physical parameter altered the variability. From what we know about AEs producing disparate latent space vectors that are not continuous and complete, this same experiment performed with a VAE would be more intuitive as toggling values from the smooth latent space distribution of VAEs makes more sense.

In a recent attempt to search for CSAGN, [40] imposed an RNN VAE (RVAE) to look for variable outliers out of a couple hundred thousand light curves from Zwicky Transient Facility (ZTF) data. Very thoughtfully, the authors acknowledged that because physical parameters can have an affect on the variability, the dataset should be balanced with respect to those parameters. Otherwise, the anolamous light curves might be anomalous with respect to the physical parameters; that is, a light curve corresponding to an AGN with anomalous black hole mass might be accounted as an anomaly in their algorithm if the balancing were not to have been done. They used two approaches to detect anomalies: an isolation forest algorithm [41]on the latent space forms of the light curves, and light curves with unusually high reconstruction errors relative to the rest of the dataset.

In addition, the authors of *Astromer* [42] employed a self-supervised irregular time series equivalent to the Transformer architecture that swaps the discrete-time positional encoding with a continuous-time positional encoding. In the equation 2.16, j is the index of the projection of the observation times that are d dimensional.

$$PE_{i,t_l} = \left\{ \begin{array}{ll} sin(t_l * \omega_i) & i \text{ is even} \\ cos(t_l * \omega_i) & i \text{ is odd} \end{array} \right\} \tag{2.16}$$

$$\omega_i = \frac{1}{100^{\frac{2i}{d_{model}}}} \tag{2.17}$$

This produces great light curve representations for downstream tasks like supervised classification, but does not teach the network to impute, forecast or capture uncertainty information. Interestingly also, the light curve representations created are 200x256 dimensions with input light curves of 200x2. (1) this is not really a bottleneck, and (2) the observational errors are not considered during training.

Moreover, *Astroconformer* [43] uses a convolutional layer with rotary positional encoding [44] and time-shifting [45] before self-attention to predict stellar surface gravity from Kepler light curves [46] as strictly a supervised task (no autoencoder or bottleneck part). They also augment Kepler light curves' noise profile and cadence to mock upcoming Rubin Observatory data and test the model's efficacy. Newer positional encoding methods like rotary positional encoding (RoPE; [44]) make it so that no 'addition' is needed to the feature vectors, and relative positional information is injected in them with clever linear algebra without disrupting the compatibility function (i.e. the scaled dot product). Time-shifting was a bit unclear as the citation for it was in reference to a github repository for a specialized RNN; the repository more accurately refers to the technique as 'Token-shift' from [47]. To our understanding, the process works by shifting feature vectors that are close or adjacent in temporal proximity back and fourth impermanently to improve the network's capturing of local temporal semantics.

Interestingly, all of the aforementioned RNN-based models are adapted to irregular time series by including Δt, the difference between observation times, as an input feature along with the observational measurements. There is no reason that continuous-time positional encoding mechanisms like RoPE or like in Astromer can not be used instead for these models too.

A very promising method called Neural Processes ([48]; NPs) has emerged as of late as an evolution of gaussian processes that is deep learning compatible. A Gaussian Process (GP) is a powerful way to ascribe probabilities to a range of functions conditioned

on data using a multivariate normal distribution through bayesian inference. For it, a covariance function or 'kernel,' κ is chosen to define a similarity measure between all pairs in the data; equivalently, the multivariate Gaussian's covariance. As such, it acts as a constraining silhouette for the possible forms the function can take. Advantages of GPs are that they are easy to update given new data and handle uncertainty very well, with a streamlined way to incorporate errors on observations. We refer the reader to an outstanding description of them in [49]. Potentially constraining aspects of GPs are that the data's fit is innately gaussian and its covariance is dictated by the assumptions we inject of the data via κ. Also, they have quite long run times during both training and inference, especially for large-scale data and adapting them to multivariate time series can be a challenge because of having to specify a positive definite covariance function. A circumvention for this is to use separable covariance functions that can be decomposed into a product of simpler one-dimensional covariance functions, but this requires the dimensions to share temporal kernel parameters.

NPs follow up with GPs by using deep learning to learn a distribution over all of the potential functions for given data example or 'context' set. They incorporate two familiar components: an encoder and a decoder. The encoder creates a local representation of each of the input-output pairs by passing each pair through an MLP. The local representations are then aggregated and passed through another MLP to form a representation of the example. There are two main subfamilies of NPs: the Conditional Neural Process (CNP; [4]) and the Latent Neural Process (LNP; [50]), which differ in so much as the decoding phase takes either the aggregated representation for the CNP or the latent variable for the LNP and passes this information along with each target input through another neural network to make the predictions. The LNP models the joint distribution of the target variables and the input variables, rather than just the conditional distribution of the target variables given the inputs, allowing it to capture complex correlations and dependencies between the variables, including nonlinear and non-Gaussian dependencies. However, the LNPF requires approximating an intractable objective function, something again familiar, ELBO. Improved variants of the NP family have come along

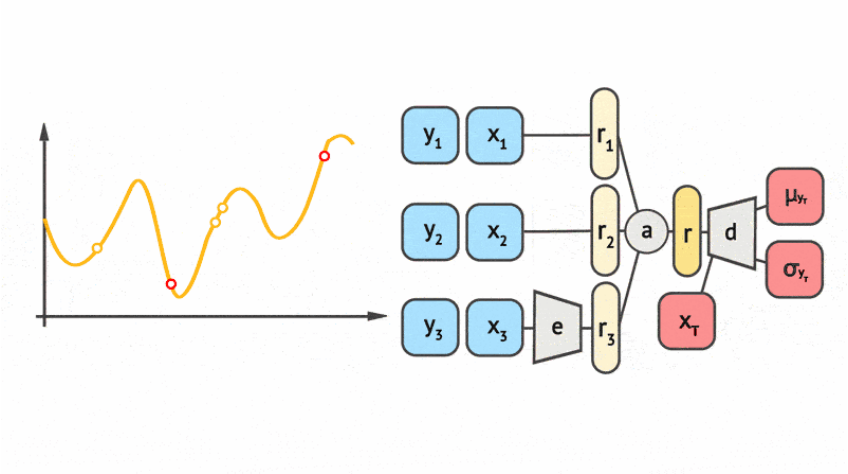


Figure 2.7: CNP depiction from [4]. Notice how each input-output pair is processed independently and the local representations are aggregated.

in the same way variants of the VAE have for both subfamilies, such as an attention variety, attnCNP and attnLNP [51] and a convolutional variety, ConvCNP [52] and ConvLNP [53]. NPs also employ self-supervised learning in their training procedure, and try to learn a function that learns over each, in our case time series, function, which is why they are sometimes referred to as meta-learners. For much further detail, we refer the reader to the following blog post [54].

The CNP has successfully been been applied to modeling Quasar variability [55], with data from All-Sky Automated Survey for Supernovae; that is, 153 objects and 150-600 points per source. Moreover, [56] used attnLNP to simultaneously infer physical parameters and probabilistically interpolate simulated multiband DRW light curves. This is the first case of considering the light curve modeling multivariately, but they were not able to apply the network to 'real' multi-band data. In general, NPs seem to require a larger number of training iterations and have more parameters than VAEs, which can increase their training time, but the potential of NPs should not be understated, and we implore the reader to keep tabs on more recent advancements to the NP family moving forward as modeling light curves as stochastic processes, as a distribution of potential functions, is a very natural approach.

For now, we look to an alternative method that has the data capacity of autoencoders, learns the data generating process in that it is a VAE, uses recent deep learning tactics like self-supervision and attention, can adapt to multivariate data, and is able to handle uncertainty like GPs and the NP family. In general, previous deep learning approaches have not necessarily been tested for the sake of reverberation mapping, where its clear use in RM would be to nonlinearly interpolate light curves before the cross-correlating steps to determine time-delays. This is in part because architectures oriented towards interpolation are few and far in between and are mainly focused on auto-encoding for feature extraction and thus for downstream tasks.

CHAPTER 3

Implementation of the Method

3.1 Heteroscedastic Temporal VAE

In describing HeTVAE, we hope that we are not being too redundant with the authors: Satya Narayan Shukla and Benjamin M. Marlin. This work is the product of Satya's doctoral dissertation [57] entitled "Deep Learning Models for Irregularly Sampled and Incomplete Time Series." HeTVAE is an evolved form of the same author's a predecessor model: The Multi-Time Attention Network (mTAN; [58]), and as such, some of the ideas slightly glossed over in HeTVAE's paper are more fleshed out mTAN's, which we hope to resolve; we also hope to slow down some of the notation for our own sake. The crucial difference between the two models is that HeTVAE more adeptly handles uncertainty quantification, but both adhere to an attention-based variational autoencoder architecture for irregular and incomplete multivariate time series and both train in a self-supervised fashion.

What is most notable about this deep learning architecture, in comparison to existing light curve modeling procedures, is its promise for using light curves in a multiband capacity

or multivariately, creating interpretable probabilistic latent space representations while probabilistically interpolating each light curve with respect to all the bands. HeTVAE applies up to date concepts from machine learning to do so: its nature as a VAE reflects current standards for generative modeling, and its use of attention and self-supervised training adhere to state of the art practices for sequential modeling.

Given that hoards of archival multiband data of time-domain surveys currently exist and there is an unprecedented data surge expected from the ground-based Vera Rubin Observatory's Legacy Survey of Space and Time (LSST), i.e. for six optical broadband filters *ugrizy* and 100 million light curves from AGN alone over the course of 10 years [59], we consider it important that data is used to its fullest extent. Additionally, the importance of considering uncertainty estimation can not be overstated if we are to use models to pre-select interesting sources and guide follow up strategies.

3.1.1 Notation

Following the authors' notation choices: in a multivariate light curve example n, there are D dimensions with each dimension d as a light curve $\mathbf{s}_{dn} = (\mathbf{t}_{dn}, \mathbf{x}_{dn})$, where $\mathbf{t}_{dn} = [t_{1dn}, ..., t_{L_{dn}dn}]$ and $\mathbf{x}_{dn} = [x_{1dn}, ..., x_{L_{dn}dn}]$ correspond to the observations taken and the number of observations L_{dn}. Moving forward, n will be dropped for clarity, and we try to use bold font to refer to vectors. We also refer to magnitudes or flux as interchangeable with $\mathbf{x}_{dn}$.

3.1.2 Time Embedding

The fundamental difference between the Transformers approach and its astronomy-adjacent, Astromer [60], is that HeTVAE does not compute self-attention. In Astromer's case, the light curve's magnitude values would be projected on to a higher dimensional plane with a chosen dimension using a feedforward layer. Then positional encoding would be injected and attention scores would be computed using self-attention. Resultingly, information is projected back to the observed points, with each observation having an estimate of how

much it should 'pay attention' to other observations in a light curve.

In contrast, HeTVAE tries to project information to a series of reference points to discretize the irregular time series. According to this concept, it instead projects the time values (and not the magnitude values) to a learned multi-dimensional plane and it also projects the time values of a series of reference points, the number of which is a hyperparameter and which are initialized as evenly spaced between 0 and 1, to a multi-dimensional plane. Henceforth, the algorithm takes attention between the projected reference time points and the projected observed time points to compute attention scores, so each reference point has an estimate of how much it should 'pay attention' to each observation in a light curve.

Ultimately there are H embedding functions, $\phi_h(t)$, that project the time points, akin to H heads in multi-head attention. ω_{ih} and α_{ih} represent learnable parameters applied to each time point in a light curve before a nonlinear sine function, akin to frequency and phase. i is the index of d_r-dimensional projection; this is shown in 3.1.

$$\phi_h(t)[i] = \left\{ \begin{array}{ll} \omega_{0h}t + \alpha_{0h}, & \text{if } i = 0 \\ sin(\omega_{ih}t + \alpha_{ih}), & \text{if } 0 < i < d_r \end{array} \right\} \tag{3.1}$$

3.1.3 Attention Mechanisms

We have seen that conceptually attention refers to how much a particular word or in this case, time point should 'pay attention' to another. In this way, the embedding created from this idea is significantly more intuitive than previous auto-encoder architectures that, for instance, would pass along information via an n-dimensional vector of the last hidden state of an recurrent cell.

The two attention-based building blocks of this neural network are coined the *value* encoding and the *intensity* encoding. The *intensity* encoding is the key stepping off point that differentiates HeTVAE from mTAN in that it provides information about the observational sparsity, or distance between points really, that is somewhat lost in mTAN's

embedding that only uses the *value* metric. This not only better informs the model generally speaking, but also makes uncertainty approximations for HeTVAE's interpolation possible.

Taking a step back, we need to define a compatibility function to assign attention weights with. In the Transformer, this is a scaled dot product between vectors. In this case we pass the time embeddings retrieved from $\phi_h(t)$ through another linear layer composed of parameter matrices $\mathbf{w}_h$ and $\mathbf{v}_h$, the same size as the time embedding, before computing the scaled dot product. Thus, each compatibility function is relative to a particular attention head h and its corresponding time embedding ϕ_h, as shown in equation 3.2. With both the nonlinear time embedding and the dense layer before the scaled dot product or compatibility calculation, the relationships or correlations found between time points can extend beyond just euclidean distance.

$$\alpha_h(t, t') = \left(\frac{\phi_h(t)\mathbf{w}_h\mathbf{v}_h^T\phi_h(t')^T}{\sqrt{d_e}}\right) \tag{3.2}$$

3.1.3.1 Value Encoding

The value encoding is computed in the 'traditional' way that attention vectors are formed in that it is a weighted sum of the *values*, where the weight assigned to each *value* is given by a compatibility function of a *query* and a corresponding *key*. In this case each observed time point's embedding acts as a *key* to the embeddings of the reference time points that are the *queries*. Each reference point then ascertains an attention weight for each of a light curve's time points, which is scaled by the observed magnitudes at those time points, as the magnitudes play the role of the *values*. Finally, those scaled magnitudes are summed to give a representation of magnitude at the reference points.

The authors initially leave it open for using alternative pooling methods to sum-based pooling (i.e. max pooling). This is because it might be advantageous to use max pooling in the case where an example is very sparse (i.e. only one or two observed points), but this is not pertinent with light curve data, so sum-based pooling will be used (and is

automatically done in the matrix multiplication operations). Moreover, the attention module is explicitly written this way by the authors to reflect parallelism with the subsequent *intensity* equation. Thus, the equation (2) from HeTVAE's paper can be simplified through the steps in 3.3 if we assume sum-based pooling.

$$\begin{aligned} val_h(t_q, \mathbf{t}_d, \mathbf{x}_d) &= \frac{pool(\exp(\alpha_h(t_q, t_{id}))\, x_{id} | t_{id} \in \mathbf{t}_d, x_{id} \in \mathbf{x}_d)}{pool(\exp(\alpha_h(t_q, t_{i'd})) | t_{i'd} \in \mathbf{t}_d)} \\ &= \frac{\sum_i^{L_d} \exp(\alpha_h(t_q, t_{id}))\, x_{id}}{\sum_{i'}^{L_d} \exp(\alpha_h(t_q, t_{i'd}))} \\ &= \sum_i^{L_d} (softmax_i(\alpha_h(t_q, t_{id})))\, x_{id} \\ &= softmax(\alpha(t_q, \mathbf{t}_d)) \cdot \mathbf{x}_d \end{aligned} \tag{3.3}$$

where $softmax_i(\mathbf{z}) = \frac{\exp z_i}{\sum_{j=1}^{K} \exp z_j} \quad for\ i = 1, 2, \ldots, K$, and its job is to normalize a vector into probabilities. Thus we have probabilities of the attention weights between a query point t_q and the observed points $\mathbf{t}_d$ for a light curve, which we take a dot product with according values of $\mathbf{x}_d$ (magnitude) to get a real-valued number that corresponds to the predicted x_d at the reference point t_q, relative to the attention head h.

For the sake of familiarity with the canonical $\text{Attention}(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$ equation [3], equation 3.3 can be viewed in its vectorized form as 3.4. This results in a vector of value encodings for each reference point $t_{iq} \in \mathbf{t}_q$ and happens to be how things are implemented in code.

$$\mathbf{val}_h(\mathbf{t}_q, \mathbf{t}_d, \mathbf{x}_d) = softmax(\alpha_h(\mathbf{t}_q, \mathbf{t}_d^T))\mathbf{x}_d \tag{3.4}$$

3.1.3.2 Intensity Encoding

The *intensity* encoding aims to inform the network of observational sparsity across time, as the *value* encoding relinquishes this information and mainly serves to put forth magnitude values to the reference points. If we did not employ this encoding, the network would not have much information to output confidence intervals at each time point we choose to project to.

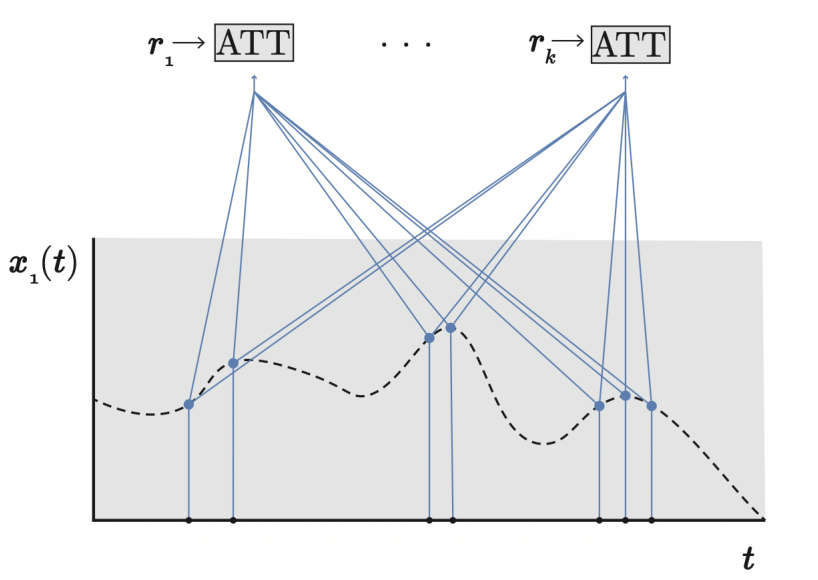


Figure 3.1: *Value* encoding. Attention is taken between a reference time point (r_k is the same as t_q) and the light curve's observed time points and $softmaxed$. The magnitude values of the observed time point that correlate most with the particular reference time point relative to a particular attention head are passed forward to the reference time point.

Attention is calculated between a reference time point and the observational time points of a particular light curve example as well as between the same reference time point and the union of all time points in the dataset, $\mathbf{t}_u$. Each set of calculations are summed and then divided, which introduces a way to propagate observational input sparsity across the network. Intuitively, if attention is large between a reference time point and a data example relative to the union of all observed time points across the dataset, there is less observational sparsity in the light curve relative to the reference point.

$$int_h(t_q, \mathbf{t}_d, \mathbf{x}_d) = \frac{pool(\exp(\alpha(t_q, t_{id}))|t_{id} \in \mathbf{t}_d)}{pool(\exp(\alpha_h(t_q, t_{i'd})|t_{i'd} \in \mathbf{t}_u)} \tag{3.5}$$

There is no strict requirement that $\mathbf{t}_u$ must be the union of all time points across the dataset although it nicely implies that the highest value the intensity encoding can be is one. If this becomes a problem with very large datasets, we can choose $\mathbf{t}_u$ to be some number (treated as a hyperparameter) of fixed points.

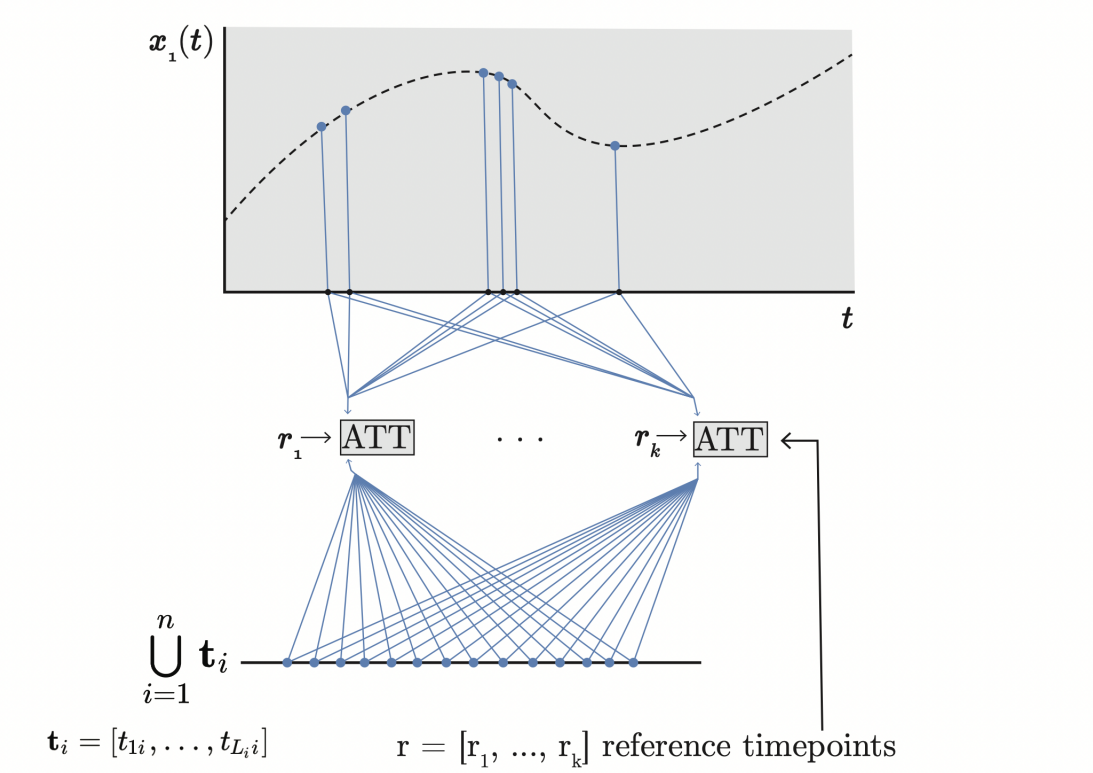


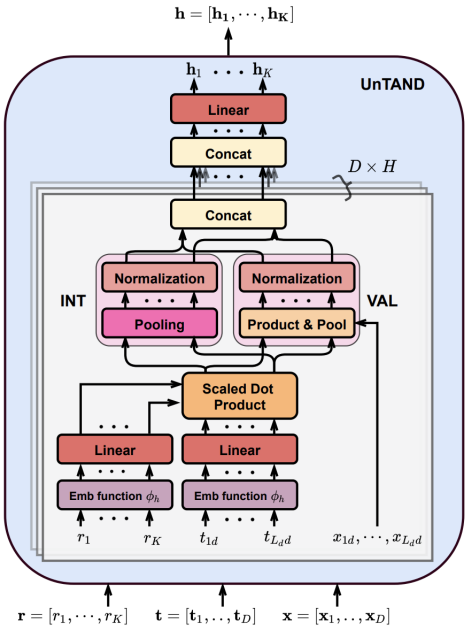
Figure 3.2: Diagram of the *intensity* encoding. The query points usually mentioned as $\mathbf{t}_q$ are the same as the reference points $\mathbf{r}$.

Sum-based pooling will be used so that (6) is equivalently

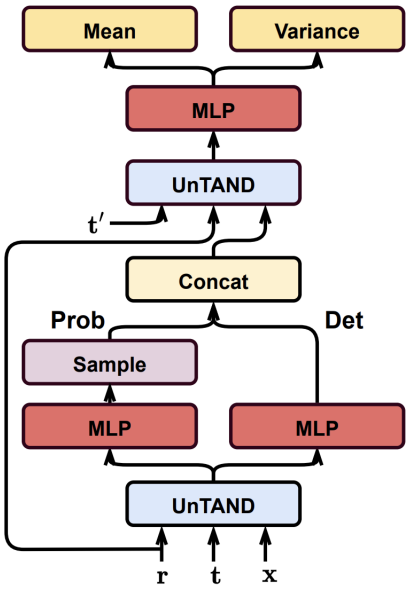
$$int_h(t_q, \mathbf{t}_d, \mathbf{x}_d) = \frac{\sum_i^{L_d} exp(\alpha(t_q, t_{id}))}{\sum_i^{L_u} exp(\alpha(t_q, t_{i'u}))} \tag{3.6}$$

3.1.4 The Rest of a Forward Pass

Once attention encodings are calculated (INT & VAL in 3.3a), they are concatenated and mixed linearly across all the attention heads and across all the data dimensions to form the intermediate representation. The intermediate representation or output of UnTAND, h, is processed through two MLPs as seen in the bottom of 3.3b. It can be seen that the model includes a deterministic pathway, outputting a discrete latent vector like a standard autoencoder, computed in parallel with the probabilistic latent state, which was shown to improve performance through their ablation studies and also which also



(a) Diagram of the entire UnTAND module.



(b) Visual for the rest of forward propagation after computing UnTAND. Note how the added parallel pathway is involved, and how the distribution is at generated at each time point.

we can turn off to make a true VAE. These pathways are then concatenated to form the final embedding of the light curve to be decoded.

$$UnTAN(t_q, \mathbf{t}, \mathbf{x})[j] = \sum_{h=1}^{H} \sum_{d=1}^{D} \begin{bmatrix} int_h(t_q, \mathbf{t}_d) \\ val_h(t_q, \mathbf{t}_d, \mathbf{x}_d) \end{bmatrix}^T \begin{bmatrix} U_{hdj}^{int} \\ U_{hdj}^{val} \end{bmatrix} \tag{3.7}$$

To decode this embedding, the series of points we wish to project the light curve to are the *queries*, t', the reference points of the embedding are the *keys* and the *values* are the embedding. Without also adding the heteroscedastic output layer to predict a distribution at each of the points, t', this model would output a constant uncertainty across time. This is the last step in figure 3.3b. In the training phase, the subsampled light curve would go through the entire forward process we have discussed, and we would try to predict to the unseen points via t', taking a loss at those predictions.

3.1.5 Objective Function

As has been shown, typical VAE loss attempts to maximize Evidence lower bound (ELBO), which serves as a lower bound for the log-likelihood of the observed data. The objective function for ELBO then consists of a negative log-likelihood term between the predicted gaussian distribution and observed value and the KL divergence term between the latent distribution and the the prior, a unit multivariate normal distribution. The authors found it helped the network to not get stuck in spurious local minima where data is thought to just be noise if a mean squared error term is added to create a composite loss because of the heteroscedastic output layer; this term is dictated by a tunable coefficient λ.

Formally, the composite loss exists from equation (11) of the original paper as $\mathcal{L}(\theta, \gamma)$, where θ is the parameters for the decoder, and γ, the encoder, and we add the coefficient β in accordance with our KL annealing schedule.

$$\mathcal{L}(\theta,\gamma) = \sum_{n=1}^{N} \frac{1}{\sum_d L_{dn}} \Bigg(\mathbb{E}_{q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)}[\log\, p_\Theta^{het}(\mathbf{x}_n|\mathbf{z}_n^{cat}, \mathbf{t}_n)] \tag{3.8}$$

$$- \beta D_{KL}(q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)||p(\mathbf{z})) \tag{3.9}$$

$$+ \lambda \mathbb{E}_{q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)}[||\mathbf{x}_n - \mu_n||_2^2] \Bigg) \tag{3.10}$$

Most notably, the loss is normalized by the number of points in an example (L_d), so it is impartial to the length or number of observations. The first two terms, 3.8 and 3.9, make up ELBO, while 3.10, the MSE term.

$$D_{KL}(q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)||p(\mathbf{z})) = \sum_{i=1}^{K} D_{KL}(q_\gamma(\mathbf{z}_i|\mathbf{r},\mathbf{s}_n)||p(\mathbf{z}_i)) \tag{3.11}$$

$$= \sum_{i=1}^{K} log\frac{\sigma_{q_\gamma i}}{\sigma_{pi}} + \frac{\sigma_{pi}^2 + (\mu_{pi} + \mu_{q_\gamma i})^2}{2\sigma_{q_\gamma i}^2} - \frac{1}{2} \tag{3.12}$$

Equation 3.12 reflects how the KL term is implemented in code relative to the author's definition in 3.11 and equation 3.14 reflects how the *nll* is calculated. x'_{jdn} is the predicted output distribution at x_{jdn}, parameterized by $\mu_{x'_{jdn}}$ and $\sigma_{x'_{jdn}}$ for the jth point for the dth dimension of an example n.

$$log\, p_\theta^{het}(\mathbf{x}_n|\mathbf{z}_n^{cat}, \mathbf{t}_n) = \sum_{d=1}^{D}\sum_{j=1}^{L_{dn}} log\, p_\theta^{het}(x_{jdn}|\mathbf{z}_n^{cat}, t_{jdn}) \tag{3.13}$$

$$= -\frac{1}{2}\sum_{d=1}^{D}\sum_{j=1}^{L_{dn}}\left(\log\sigma^2_{x'_{jdn}} + \frac{(x_{jdn}-\mu_{x'_{jdn}})^2}{\sigma^2_{x'_{jdn}}} + \log(2\pi)\right) \tag{3.14}$$

3.2 Adaptation

The author's code base, `https://github.com/reml-lab/HeTVAE`, provided us with the model schematic, but was not without its faults. For instance, all of their experiments used one attention head, but when we tried to use several, we found a bug in the multi-head implementation that needed fixing. Some of the other bugs that we can recall fixing were in their implementation of dropout and in the propagation of the device variable used when working with the model (cpu/gpu/mps). Otherwise, we wrote software to more conveniently prepare datasets, save and load model checkpoints, visualize training and latent spaces, etc, which can be found at `https://github.com/mwl10/hetast/tree/master/src`. All jupyter notebooks that reflect the code procedures we used for our experiments are included in this repository as well, and explanations for the dataset class we wrote for more convenient preprocessing can be found in appendix A.

3.2.1 ZTF Data & Preprocessing

The Zwicky Transient Facility (ZTF) is located at the Palomar Observatory and is a computer-automated time-domain survey that uses a 4-foot Schmidt telescope with with a 47^2 degree-wide field of view camera, scanning the entire Northern sky with an approximately three-night cadence for phase I (May 2018 – September 2020) and a two-night

cadence for its phase II (December 2020 – present) for its custom g and r filters and a less resolved four-night cadence for its i filter ([61] [62] [63]).

The ZTF api was queried using a function and sample catalog with 3398 objects generously provided by Paula Sánchez Sáez to obtain light curve files for each of the g, r and i bands from the latest data release 15, which had a total observation span from March 2018 – November 2022 or MJD 58,200– 59,700 (for i: 58,200-59,200). Physical properties like black hole masses, redshifts, Eddington ratios and luminosities were included in the catalog we were given, but are from the catalog of Spectral Properties of Quasars from Sloan Digital Sky Survey Data Release 14 [64].

We added light curves from data release 16 which was further through Jan 2023 or 58,200–59,900 (for i: 58,200-59,400) from nine objects with known lags in the ZTF-relevant bands garnered from higher resolution surveys to test our method for reverberation mapping. These objects are: 3C120, 3C273, H2106-099, MCG+08-11-011, Mrk 142, Mrk 817, Mrk 876, NGC 2617 and NGC 5548. The api returned 3407 raw light curve files (many without any observations) for each of the bands.

We filtered the light curves as guided by [40], who help delineate nuances of the ZTF api. The authors note that it occasionally returns more than one light curve for one band so it would be apt to choose the longest of those returned. They go on to only retain light curves with mean magnitudes brighter than 20.6 as per this value being close to the limiting magnitude of ZTF and fainter than 13.5 to avoid saturated observations. We retain light curves according to the limiting magnitudes for each specific band. The limiting magnitude for $g = 20.8$ mag, $r = 20.6$ and $i = 19.9$ [61]. [40] sensibility set the minimum mean magnitude to be 20.6 for the g band light curves they worked with, which is what we based our filtering on. In light of this, we set the minimum allowed $\overline{mag}$ for g to 20.6, r to 20.4 and i to 19.7. Additionally, they removed sources with ranges less than 730 days and less than 50 observations, and sources that did not show variation based on two calculated variability features, P_{var}, the probability of a source being intrinsically

variable and σ^2_{NXS}, the normalized excess variance. Futhermore, the ZTF documentation mentions to remove points with any **catflags** quality score unequal to 0, and we average the magnitude values for any duplicate time points.

We decided to remove the constrictions for number of observations entirely, allowing for light curves with greater than one observation to be considered, and we removed our constraint on variability. The objective function normalizes the loss from each light curve over the number of points it has so removing the constraint on the number of observations should not interrupt the network very much. As for removing the variability constraint, we hypothesize that it will let the network learn to separate light curves with regard to their 'amount' of variability in the latent space, apart from including them to have a larger representative distribution of AGN variability. Most importantly, for the multivariate models, we made it so that the the model could consider examples with missing light curves in any of the bands (so long as it was not all three of them of course). To do this, we had to make some extra accommodations in dataset preprocessing for the missing light curves and zero out both the subsample mask and reconstruction mask for them during training so they would not be considered in the learning process. This increases the model's ability to ingest uglier, more incomplete datasets.

3.2.1.1 Outlier Pruning

As for cleaning the light curves, points with magnitude error greater than 1 are removed. Then we remove points further than 10 standard deviations from the mean magnitude of the light curve (if any). We initially implemented code to follow the likes of [65] to remove outliers, in which a three-point median filter is applied to the time series and a quintic polynomial is fit whereby points with residuals from this higher than 0.25 mag are removed unless more than 10 percent of the light curve points are removed; otherwise the clipping threshold is increased until that is not the case. We instead figured the network should be acknowledging unusual variation and activity and learning from it, instead of trying to smooth it out.

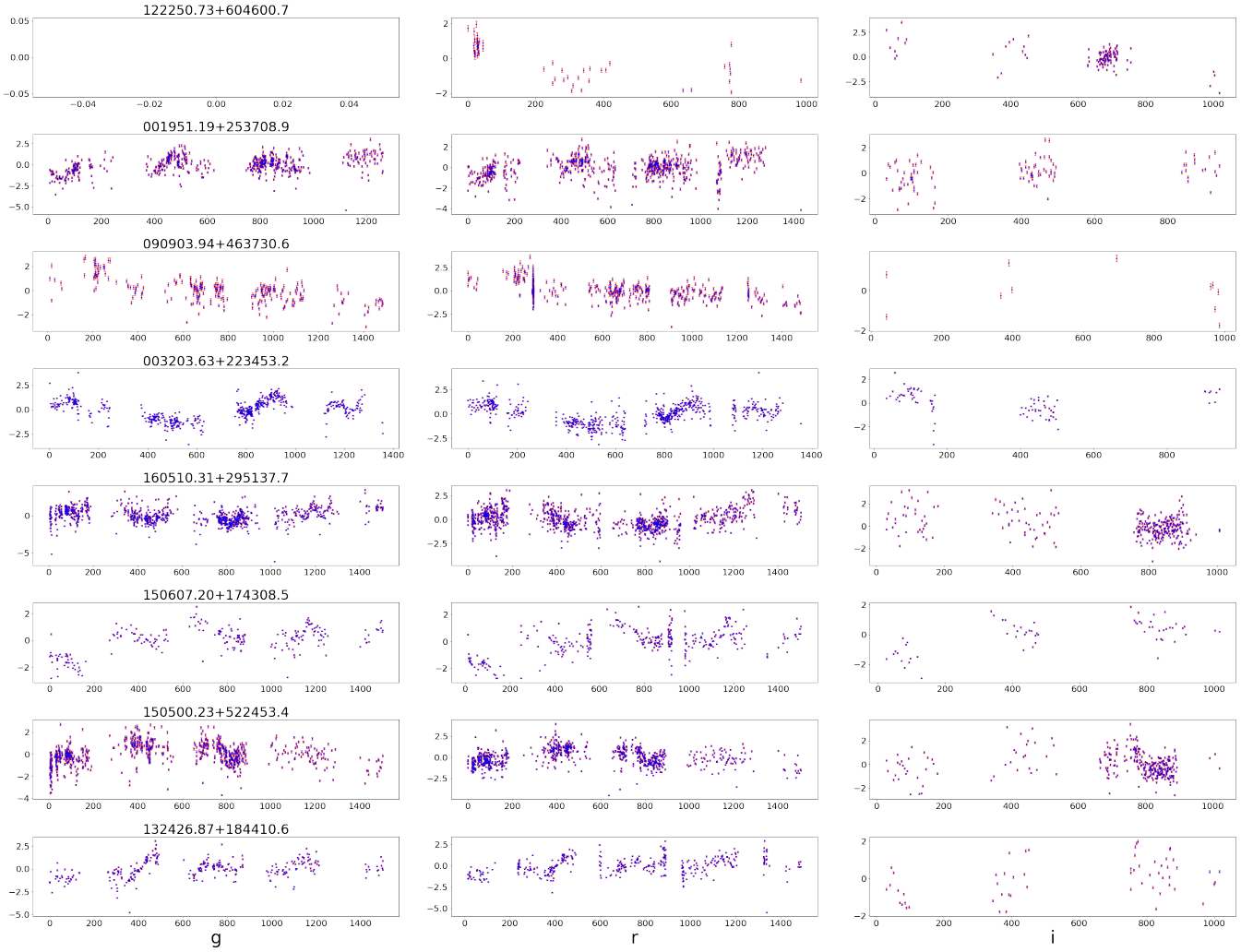


Figure 3.4: Preview of cleaned and normalized ZTF light curves of eight arbitrary objects in the dataset.

We found that chopping-off light curve values past certain time points could be beneficial if the distribution of the ranges of the light curves was right-skewed. As such, we added a function to remove points past a given number of standard deviations beyond the mean of the ranges of the light curves across the dataset. This is because the references points are laid out by the network according to the light curve ranges. We did not end up needing to do any 'cutting' because there was no right-skewness, but rather left skewness in our data; a large majority of the light curves had a range of 1500 days, so the network will try to spread out the reference time points according to this range.

3.2.1.2 Normalization

Magnitude values are normalized for each individual light curve to have a mean of zero and a standard deviation of one after the outliers have been removed. The magnitude errors are normalized by dividing them by the standard deviation of the light curve's magnitudes. For a while we had it so that each light curve subtracted its own minimum time value, which was a subtle bug that broke multivariate training. Instead, the time points of each light curve are subtracted by the minimum time value across all dimensions for an example/object. We found that normalizing the time values in a min-max capacity (to between 0 and 1) hurt performance, which intuitively makes sense as normalizing time values this way would corrupt the time-scale dependencies and variability relationships in time that the model is trying to learn, generally interrupting the physical interpretation of time. On the other hand, dividing time values by 365 produced no effect so we withheld from doing this as well. We show the normalized light curves from a handful of arbitrary objects in 3.4.

3.2.1.3 Light Curve Property Distributions

From 3.6 we can see that apart from the uniformly-distributed redshift, the distributions of the AGN physical properties are Gaussian. The number of observations and the median cadence plots confirms that i very much less sampled relative to the other bands, and the ranges generally align with the time frames of the data release, while the median values of σ^2_{nxs} confirm our understanding that variance increases across lessening wavelength.

dataset	min length	shape	max time
gri	0	(3177, 3, 4396, 3)	1687.1758
	25	(1527, 3, 4396, 3)	‘’
	50	(1033, 3, 4396, 3)	‘’
gr	0	$(3167, 2, 3950, 3)$	1687.1758
	25	(2815, 2, 3950, 3)	‘’
	50	(2668, 2, 3950, 3)	‘’
gi	0	$(3106, 2, 2408, 3)$	1687.1367
	25	(1527, 2, 2408, 3)	‘’
	50	(1034, 2, 2408, 3)	‘’
g	0	$(3087, 1, 1962, 3)$	1687.1367
	25	(2831, 1, 1962, 3)	‘’
	50	(2690, 1, 1962, 3)	‘’
r	0	$(3141, 1, 1988, 3)$	1505.0703
	25	(2956, 1, 1988, 3)	‘’
	50	(2778, 1, 1988, 3)	‘’
i	0	$(2613, 1, 539, 3)$	1002.1328
	25	(1537, 1, 539, 3)	‘’
	50	(1039, 1, 539, 3)	‘’

Table 3.1: Dataset shapes given a particular filter on the minimum number of observations allowed. A dataset that is of the dimensions (1527, 2, 2408, 3) means that there are 1527 examples with 2 filter dimensions/light curves per example. 2408 would be the number of observations (across both of the light curves) for the longest/most observed example in the dataset. Each example specifically formats its observations (time/mag/magerr) across its own two light curves without regard for the other examples and then pads itself with zeros to match 2408. We better describe how the formatting of the datasets is done in appendix A (figure A.7) as we acknowledge this is not very clear.

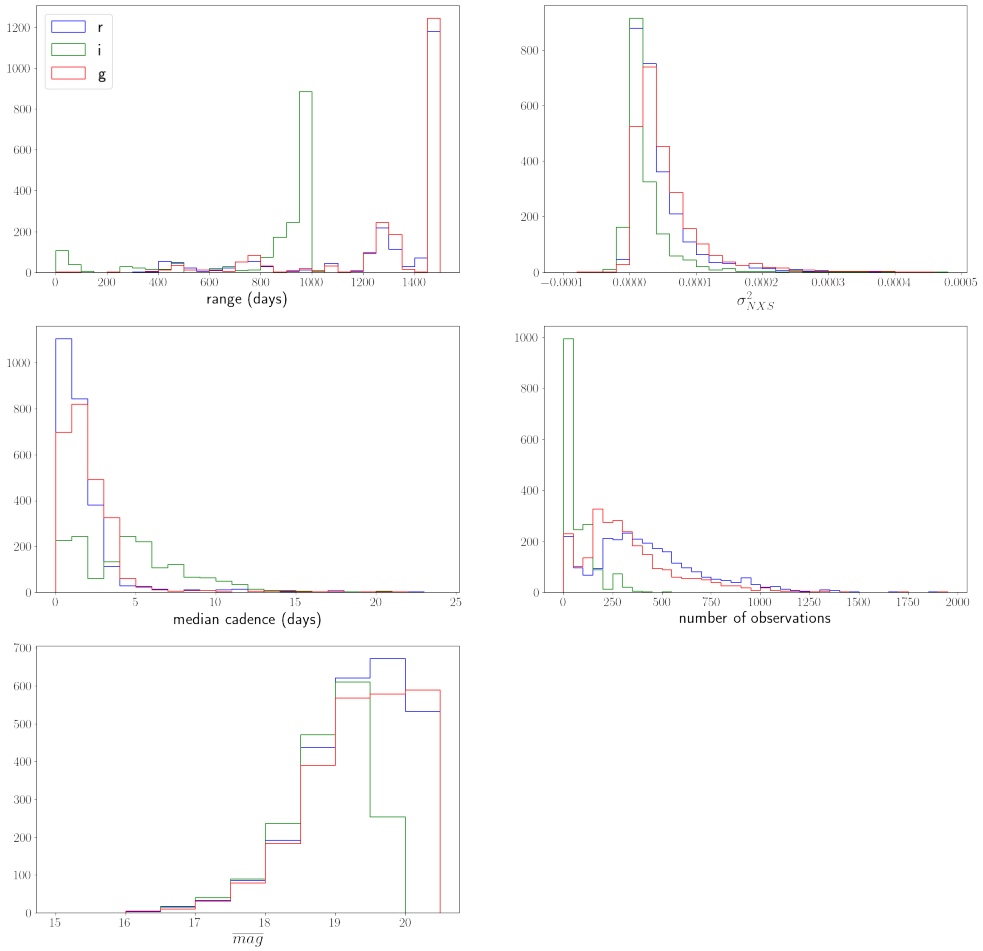


Figure 3.5: Light curve property distributions from the sample catalog.

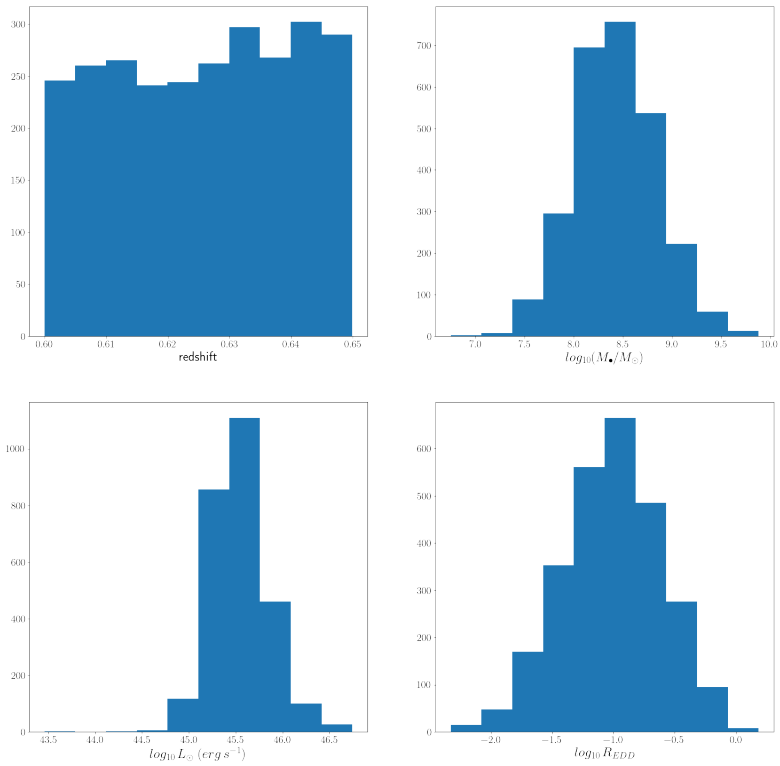


Figure 3.6: Physical property distributions from objects in the sample catalog.

3.2.2 Pre-training Configuration

All of the subsequent hyperparameter tuning was done using a validation set that is a 20% subset of the training set. We use an overlapping validation set because each subsample of an object's light curves is distinct and to follow the precedent set by HeTVAE's authors. In referring to different models trained on different datasets univariate or otherwise, we will refer to it by the ZTF band(s) it was trained on, i.e. *gri* model; *g* model; *gr* model. We specifically tuned on the *g* band data, finding good results across both univariate cases of the other bands and multivariate cases. We have set up the hyperparameters and other network configurations to be read using Hydra from a *.yaml* configuration file so that we can change the configuration of the network and run studies to obtain optimal hyperparameters values on the command line.

3.2.2.1 KL Annealing

A common problem with training VAEs is that the KL divergence term in the loss vanishes, which results in a lack of learned structure in the latent space. The latent vectors immediately collapse into the shape of prior distribution, throttling the ability of the

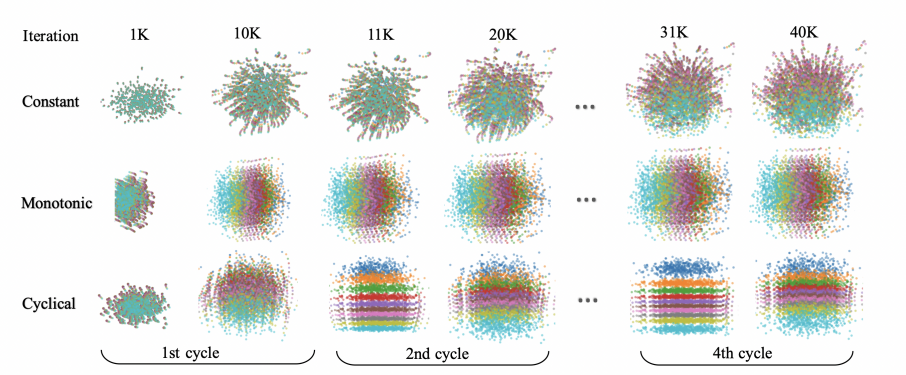


Figure 3.7: A monotonic annealing schedule means that the coefficient for the KL term is gradually increased until it reaches one, staying there. A constant schedule implies the coefficient is always one. This figure from [5] shows the learned space structure for a 10 sequence dataset trained on an LSTM VAE with each sequence as a 10-dimensional one-hot vector with the hot-one appearing in each different position.

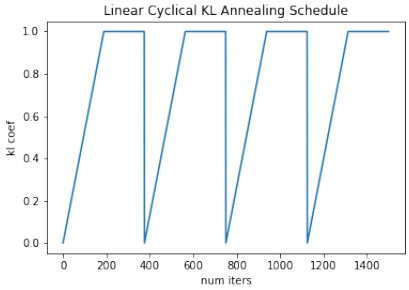


Figure 3.8: An example schedule where the time that KL is increasing versus constant is split 50-50 across a given number of iterations with 4 cycles in total per 1500 iterations.

network to learn a more interesting posterior. At the same time, because the network ignores or blocks the now vanished KL term, the network will operate like a standard AE and focus more on the reconstructions instead of global information. A simple way to mitigate this issue is KL Annealing [5], in which the coefficient of the KL loss term β is cyclically increased. Thus the network will learn structure, gradually condense this structure to the prior and rinse and repeat.

3.2.2.2 Handling Errors on Input

In the case of having errors on the observations, as are generally available in astronomy, a mean squared error loss can be weighted such that the reconstruction error is punished more where measurement errors are small and less when measurement error is large.

Equation 3.15 shows this augmentation, which has been used on several occasions ([39], [38], [40]) in modeling light curves with deep learning. We keep aligned with the notation of HeTVAE's authors such that x' represents network's predicted values and x and σ_x^2, the true values. Specifically for VAEs, the MSE loss must also be rescaled by a factor (F_1) after it is weighted by the errors so it does not overwhelm the KL term.

In terms of augmenting HeTVAE's loss function, which we detail in section 3.1.4, to account for observational errors during training, it is less trivial with the additional negative log-likelihood term from the heteroscedastic output layer. We have to weight this term as well as the MSE term, and rescale them both to keep the balance with each other, as well as the KL term. We have implemented the following adjustment to weight $\mathcal{L}_{nll}$ in equation 3.16, but we have not yet used it for training (with F_2 as the scaling factor for the negative log-likelihood). We want to maintain the 5:1 MSE to NLL ratio and just about 1:1 MSE to KL ratio; as such, we set F_1 to 10 and F_2 to 100.

$$\begin{aligned} L_{MSE} &= \mathbb{E}_{q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)}||\mathbf{x}_n - \mu_n||_2^2] \\ &= \sum_{d=1}^{D}\sum_{j=1}^{L_{dn}}(x_{jdn} - \mu_{x'_{jdn}})^2 \\ L_{WMSE} &= \frac{1}{F_1}\sum_{d=1}^{D}\sum_{j=1}^{L_{dn}}\frac{(x_{jdn} - \mu_{x'_{jdn}})^2}{\sigma^2_{x_{jdn}}} \end{aligned} \tag{3.15}$$

$$\begin{aligned} L_{nll} &= log\, p_\theta^{het}(\mathbf{x}_n|\mathbf{z}_n^{cat}, \mathbf{t}_n) \\ &= \sum_{d=1}^{D}\sum_{j=1}^{L_{dn}} log\, p_\theta^{het}(x_{jdn}|\mathbf{z}_n^{cat}, t_{jdn}) \\ &= -\frac{1}{2}\sum_{d=1}^{D}\sum_{j=1}^{L_{dn}}\left(\log\sigma^2_{x'_{jdn}} + \frac{(x_{jdn} - \mu_{x'_{jdn}})^2}{\sigma^2_{x'_{jdn}}} + \log 2\pi\right) \\ L_{wnll} &= -\frac{1}{2}\frac{1}{F_2}\sum_{d=1}^{D}\sum_{j=1}^{L_{dn}}\left(\log(\sigma^2_{x'_{jdn}} + \sigma^2_{x_{jdn}}) + \frac{(x_{jdn} - \mu_{x'_{jdn}})^2}{\sigma^2_{x'_{jdn}} + \sigma^2_{x_{jdn}}} + \log 2\pi\right) \end{aligned} \tag{3.16}$$

We also have implemented a Monte Carlo approach to incorporating uncertainty on the

observations by sampling from Gaussian noise on par with the estimated measurement error at each time step create new-examples, i.e. resampling the light curves.

$$x_{dn} = x_{1dn} + \epsilon_{1dn}, x_{2dn} + \epsilon_{2dn}, ..., x_{L_{dn}} + \epsilon_{L_{dn}} \ ; \ \epsilon_{jdn} \sim \mathcal{N}(0, \sigma_{jdn}) \tag{3.17}$$

3.2.2.3 Fraction

The network trains itself by imputing a fraction of the observed points of a light curve and trying to predict to the unseen points with each training iteration taking a different random subsample (as shown in figure 3.9). This fraction is yet another hyperparameter, acting as the self-supervised aspect of the neural network. From intuition, using a higher fraction means the network is learning to impute smaller and smaller segments of the light curve. Too little of a fraction and too much is being asked of the network so the loss landscape might be impeded. The authors used 0.5 for this value in all of their experiments and so do we. It would be interesting to experiment more with the frac hyperparameter in future work. If warm starts were used on the frac hyperparameter, would the network be able to interpolate larger and larger gaps and thus 'better' learn the dataset; or would this be more memorization than we would like?

3.2.2.4 Model Size

Tuning the hyperparameters that dictate the size of the model is difficult because they are all interconnected. They are six: $latent\_dim$, rec_hidden, num_ref_points, $embed\_time$, $width$ and num_heads. The first, embed time, is the dimensionality of the time embeddings. The second, the dimensionality of the output of the UnTAND module before projecting to the latent space, the projection being two linear layers with a size equal to $width$. $latent\_dim$ is the dimensionality of the latent space, with a $latent\_dim$-dim vector per reference time point in the latent space. num_ref_points is the number of reference time points. The number of attention heads is num_heads.

We suspect the ratio between each of these hyper parameter is the most important aspect of them, so that none bottleneck one another or constrain the network's ability to learn,

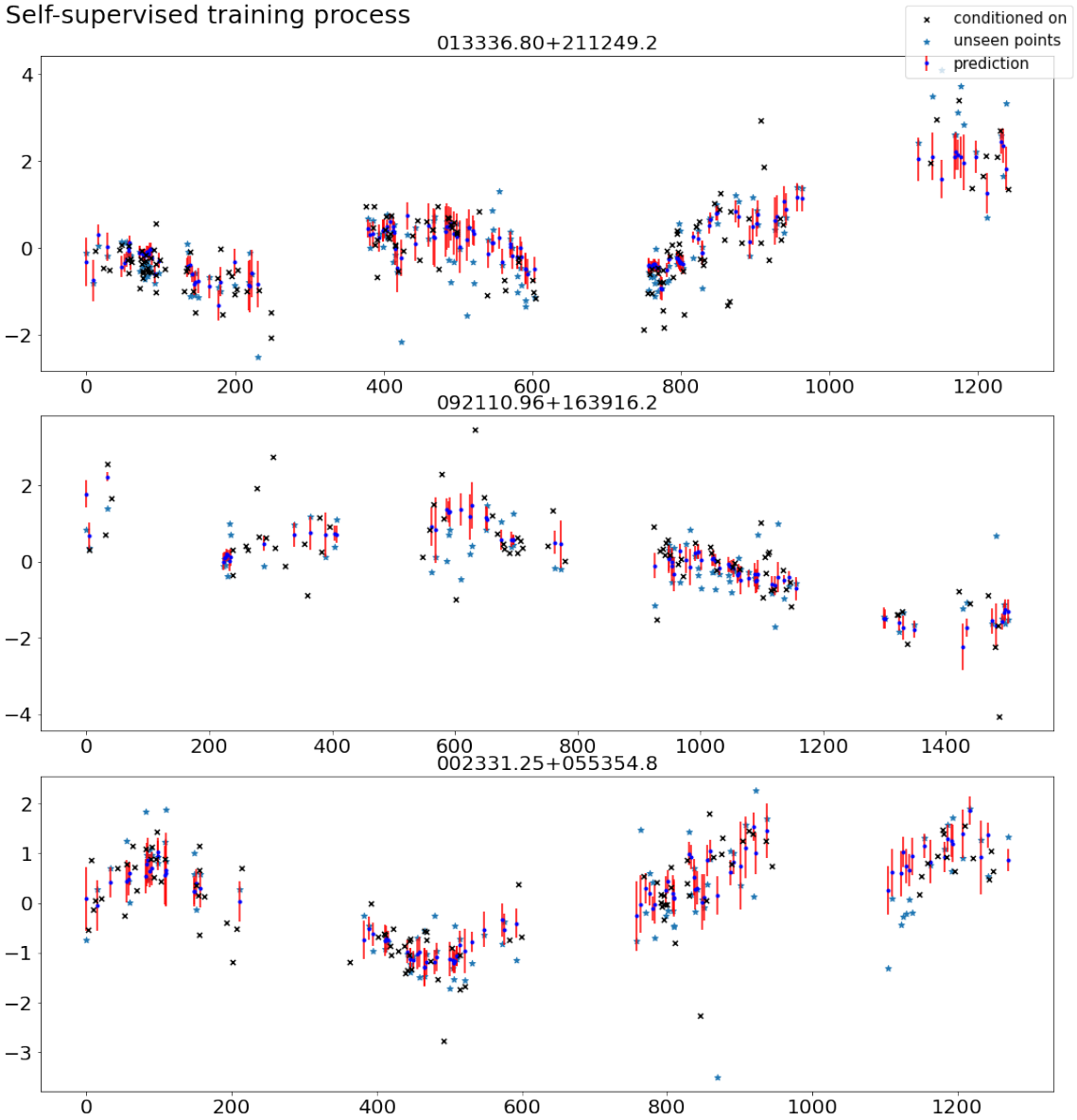


Figure 3.9: Visualization for the network's training procedure. A random subset of half of the points are given to the network and it attempts to make predictions at all the unseen observations.

i.e. if *embed_time* is too small then how does that limit the usefulness of having a larger and larger latent space? Does increasing any one of the hyperparameters 2-fold really increase the capacity of the network or do the sizes of them work more in unison?

Given that there are six hyperparameters associated with the model size, that would entail searching over a very large space of combinations, while having to tune nuisance parameters for each different model size trial so it is fairly contested. An example of a nuisance parameter would be the learning rate because its optimal value changes with the model size. Instead, the approach we took was to appreciate the heavy lifting that the authors did hyperparameter tuning on their datasets, finding that the model size hyperparameters were pretty generally applicable and robust with respect to different datasets.

The first of the authors' datasets, *PhysioNet*, is a 37-dimensional time series dataset with measurements spread across 48 hours from intensive care unit (ICU) records with 8000 examples. The largest number of observations across all the 37-dimensional examples is 203, and the union of all the time points across the dataset is 2880. Thus, in our formatting scheme the dataset shape would be $(8000, 37, 203, 2)$. The second, MIMIC-III [66], is a privately-accessed dataset of irregularly sampled physiological signals across time from the Beth Israel Deaconess Medical Center., but there are preprocessing details in appendix A.1.1 of [67]. In it, there are $53,211$ examples with 12 physiological variables and 48 hours of data that all take all less than 1 sample per hour. Without requesting access to this dataset, we can thus infer that, at most, the largest number of observations across all the 12-dimensional examples is $12 * 48 = 504$ because each dimension of the 12 would potentially a maximum of 48 observations, which would create a dataset roughly of the shape $(53211, 12, 504, 2)$. As compared to the ZTF data and light curve datasets generally, these datasets have more dimensions, while the ZTF data has many more observations. The complexity of their datasets versus the light curves for the model to learn hopefully balances out by way of this tradeoff.

dataset	num-heads	num-ref-points	embed-time	rec-hidden	latent-dim	width
Physionet	1	16	128	128	128	128
MIMIC-III	1	16	128	128	128	512
ZTF	16	16	128	128	128	512

Table 3.2: Model size hyperparameter choices across datasets.

For the hyperparameters that changed across their datasets, we performed a grid search for the optimum in our dataset, i.e. for the *width* of the linear layers as this was 128 for *PhysioNet* and 512 for MIMIC-III. We also tested configurations ad hoc, like for instance it seemed as though only having 16 reference points could potentially be constraining, but saw no obvious improvement with increases of this value to 32 and 64. Also, as discussed previously, there was a bug in their model that precluded us being able to use more than one attention head so upon fixing this bug, we performed a grid search on the number of attention heads and found the optimum to be a quite contrastive 16 as compared with the other datasets' 1. 3.2 shows the hyperparameter values across datasets, including those tentatively chosen for the ZTF dataset.

We seemed to be close to the 1-gpu memory limit with this size model too, restraining us from trying larger models. The final model contains 574662 learnable parameters and each model checkpoint rounded out to about 6.72 megabytes. We can cautiously hypothesize that more substantial increases in the models capacity would require maintaining the ratio between model size hyperparameters because of their working in tandem, in effect, doubling the size of the network. In the future, more rigorously tuning of the hyperparameters would likely be a constructive effort.

3.2.2.5 Union Time-points

In the HeTVAE paper, authors left experimenting with different arrays for the intensity attention calculation for future work. They used the union of all the time points across the dataset, $\mathbf{t}_u$, which works for datasets where observations might not be always taken, but if they are taken, are taken on regular interval. For example, if across all the time

series, observations can be taken at any value [0, 0.5, 1, 1.5, ... ,100], which would make a 'union tp' of length 200. It also is reasonable if time values are rounded and thus there are less measurement times, which is the approach taken with the PhysioNet Challenge 2012 [68] (rounded to the nearest minute from 48 hours total) and which resulted in 2880 potential measurement times. For our ZTF dataset, the union of all the recorded time values greater than $200,000$, making this unreasonable and unnecessary to use and computationally expensive. Instead we choose an array whose length is a hyperparameter, evenly spread across time between the maximum and minimum time values of the normalized dataset. Accordingly, we tried a grid search between lengths of 500 to 5000 by increments of 500 and landed on 3500.

3.2.2.6 Batch Size

Frequently, we ran into memory issues with GPUs when using Durham's Nvidia Computing Cluster (NCC), likely because of a lack of experience with debugging these issues and training these networks at scale. Because of this, we ended up with very small batch sizes of 2, which does nothing harmful other than to slow down training. We tried tactics like using float16 tensors (half tensors) for the dataset, but PyTorch does not have most operations implemented on such size tensors. For reference, total dataset size for all three bands is 0.39 GB.

3.2.2.7 Making Predictions

HeTVAE has a number of ways to express uncertainty as we have seen. (1) it can vary the latent state distribution for each example, resulting in a different interpolation. (2) it expresses uncertainty at its heteroscedastic output layer, predicting a marginal distribution at each time point we interpolate to. Therefore when making interpolations we take 10 samples from the the latent state distribution, decode them separately to get an interpolation for each sample and average them to get a final result. When sampling from the latent space for our anomaly detection, z from qz, we average 10 latent space samples.

3.2.3 Experiments

The intention for chapter four is to give a 'sampler' of HeTVAE's capabilities as well as including more of what we have learned about training and applying the model. We will show several different model iterations for both multivariate and univariate data cases and detail failures, room for improvement and successes in applying those models to various tasks. The potential usages for our latent variable model are ranged, from interpretability experiments, to anomaly detection or filtering, to forecasting, to downstream tasks with both the latent space (setting up supervised problems) and interpolations (reverberation mapping).

3.2.3.1 Filtering Algorithm

Luckily we have a smooth distribution over the latent space to work with, so outliers from the general population are isolated conveniently. The issue is that these outliers can be vague outliers; that is, they are outliers with respect to any trait that makes them 'different' from the rest of the population. One way to cancel out traits that we do not want to appear as anomalous is by balancing the dataset with respect to that trait, and thus outliers with respect to all leftover unbalanced traits are realized. For instance, if we wanted to balance the dataset with respect to the number of observations, we could do this by keeping a finite number of light curves for bins that each correspond to a different range of epochs, i.e. we can have a maximum of 10 light curves with 0-10 observations, a maximum of 10 light curves with 10-20 observations and so on. This example in particular should be unnecessary for our model, because the objective function normalizes number of observations L_{dn} out of the calculated error. Hence also, we can adjust the loss function to make the model agnostic to a specific trait.

Interestingly, the binning improves the model's accuracy for examples that were previously anonymous with respect to that trait, at the expense of losing training data. An analogous example would be if we were trying to predict a rare cancer from medical images. It would be likely that the dataset only has a very small proportion of examples that constitute cancer so the model could achieve an impressive average accuracy if it just

predicts that the all the images do not constitute cancer. We could balance the dataset to get an even number of cancer examples and non-cancer examples to train on or we could adjust the loss function so that there is more of a penalty for the network to be wrong with examples of cancer (relative to their proportionality in the dataset) to the same end. But rather than taking the direction of 'evening out' subgroup's, and having anomalous activity with respect to all other aspects of, in this case Quasar variability, 'pour out,' we decided to try and find ways to target specific types of variability with the latent space.

To get a general idea of what is happening in the latent space embedding vectors of the light curves, we can use PCA visualizations; PCA preserves global structure by finding a lower dimensional hyperplane that accounts for the most variance in the data, where the first principal component explains the most variance, the second (orthogonal to the first) the next highest variance, etc. Each principal component, that explains a given amount of variance in the data, has an amount of contribution from each of the original features, so if some of these original features explain little to no variance in the data, they are 'unnecessary,' and hence principal components are often used to remove redundant and unnecessary features input into a learning algorithm. For our case, what PCA can do is identify latent space features that account for major differences in variability patterns of the light curves that the network has intuited, akin to drawing the longest lines across the high-dimensional latent space distribution and placing the light curve embeddings along the line. Then, we can place object's embeddings into bins across the said line, average the embeddings, and reconstruct these averages to see how the light curves look *different* with respect to each bin of the component.

Indeed, each principal component will also highlight a relatively different aspect of variability that changes across the dataset, so if the variability that corresponds to a particular bin, from a particular principal component, is interesting, we can take a look at all the light curves that exist in such bin already and in the future, we can encode any light curves with the network and see if their encoding winds up in this bin as a filtering

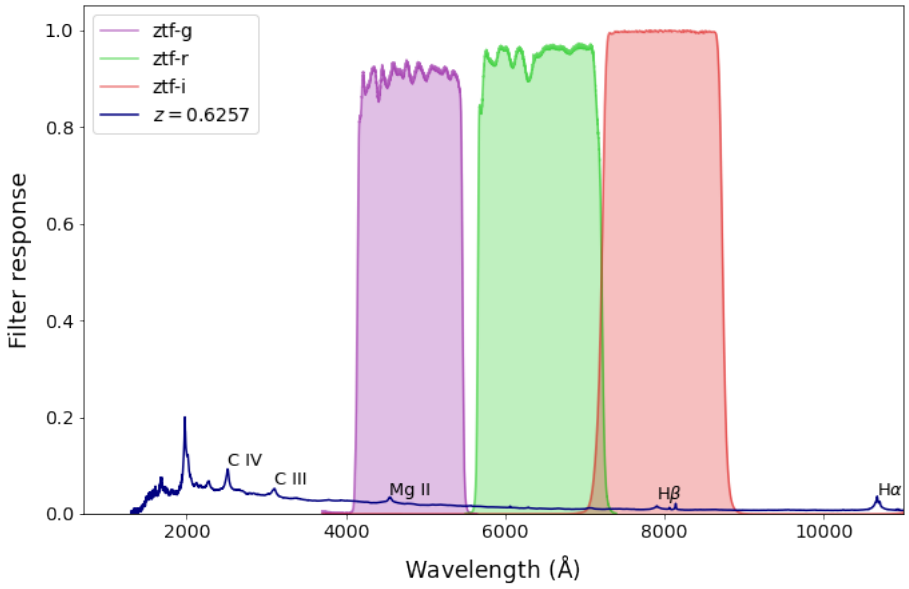


Figure 3.10: This visualization of superimposed broadband filters for ZTF at $z = 0.6257$ is from code written by dr Anđelka Kovačević and Isidora Jankov [6].

algorithm. It may be that one of these principal components is reflective of variability metrics like excess variance or fractional variability, or a step further, is evidence of flaring activity or abrupt decreases/increases in flux that we think are indicative of CSAGNs.

We also aim to look at the traditional ways of detecting anomalies with the autoencoder like searching for abnormally large reconstruction error or passing all of the latent space features through an IF [69]. Briefly, an IF is based on a decision tree, which is set up by selecting random features from the data and creating thresholds between the minimum and maximum values for those features in the dataset. Then, anomalous activity can be sussed out because it tends to take less time for the anomaly to be isolated by the tree.

3.2.3.2 Reverberation Mapping

We plan to use HeTVAE's interpolations of the ZTF light curves to higher cadence, regularly spaced intervals, in tandem with PyROA [29] in an attempt to improve time-delay estimates. As stated previously, we test this proposed method on objects with known reference lags garnered from other surveys, but reduce those mentioned to specifically the

AGNs 3C273, MCG+08-11-011 and NGC 5548 in our analysis. We first are curious as to whether we can reproduce the original lags using the ZTF light curves of these objects, knowing that the sampling intervals of ZTF are on par with or (more likely) longer than the time delays from the objects we have chosen, which will surely have an effect on the ability to estimate time delays. We use the quite powerful and unassuming framework *PyROA* [29] for this. In the case that we are able to reproduce the lags with the ZTF light curves and PyROA, it will be informative if interpolating the light curves with HeTVAE interrupts the time delay estimations or reaffirms them. Alternatively, if it is the case that PyROA can not reproduce the lags, but in using the interpolations of HeTVAE, we can reproduce or even constrain them, the algorithm's helpfulness is shown. Even with light curves that have clear variable features for PyROA to anchor onto, because most of the lags are shorter than the cadence, a higher resolution of interpolation from HeTVAE may assist, and with light curves that have missing features from a lack of observations, HeTVAE may be able to infer these features with its semantic knowledge from the general population of light curves or, if a multivariate model, may be able to from the other light curve dimensions.

We can presume that the network attempts to learn generic variability features, taking semantic knowledge across the dataset to create the reconstruction/interpolation, so its ability to account for highly-resolved, individual shapes might be lacking somewhat. Thus, if it is the case that the network's interpolations are not precise enough, we can move to fine-tune the model to a smaller subset of light curves or, if necessary, particular segments of those light curves.

Table 3.3 shows the estimated lags for the objects with the associated references listed as well. In most instances the lags were measured assuming a different filter than g as the driving light curve so we subtracted the g lag from them as per its being the shortest wavelength light curve available with ZTF data, and thereby is used as the driving light curve. In the following analysis, we delve into the epoch separations for each object's light curves to prepare them for PyROA's lag estimation. We try to match the detrending

Object	τ_{gg}	τ_{gr}	τ_{gi}	cite
MCG+08-11-011	$0.00^{+0.08}_{-0.07}$	$0.69^{+0.16}_{-0.15}$	$1.02^{+0.20}_{-0.18}$	[72]
3C 273	$0.0^{+1.77}_{-1.96}$	$11.01^{+1.71}_{-2.07}$	$13.94^{+2.94}_{-2.37}$	[73]
NGC 5548	$0.0^{+0.06}_{-0.04}$	$0.93^{+0.06}_{-0.07}$	$2.01^{+0.11}_{-0.08}$	[74]

Table 3.3: Known lags for the AGN MCG+08-11-011, 3C 273 and NGC 5548. Lags for 3C 273 were estimated with PyROA, while the other two with Javelin.

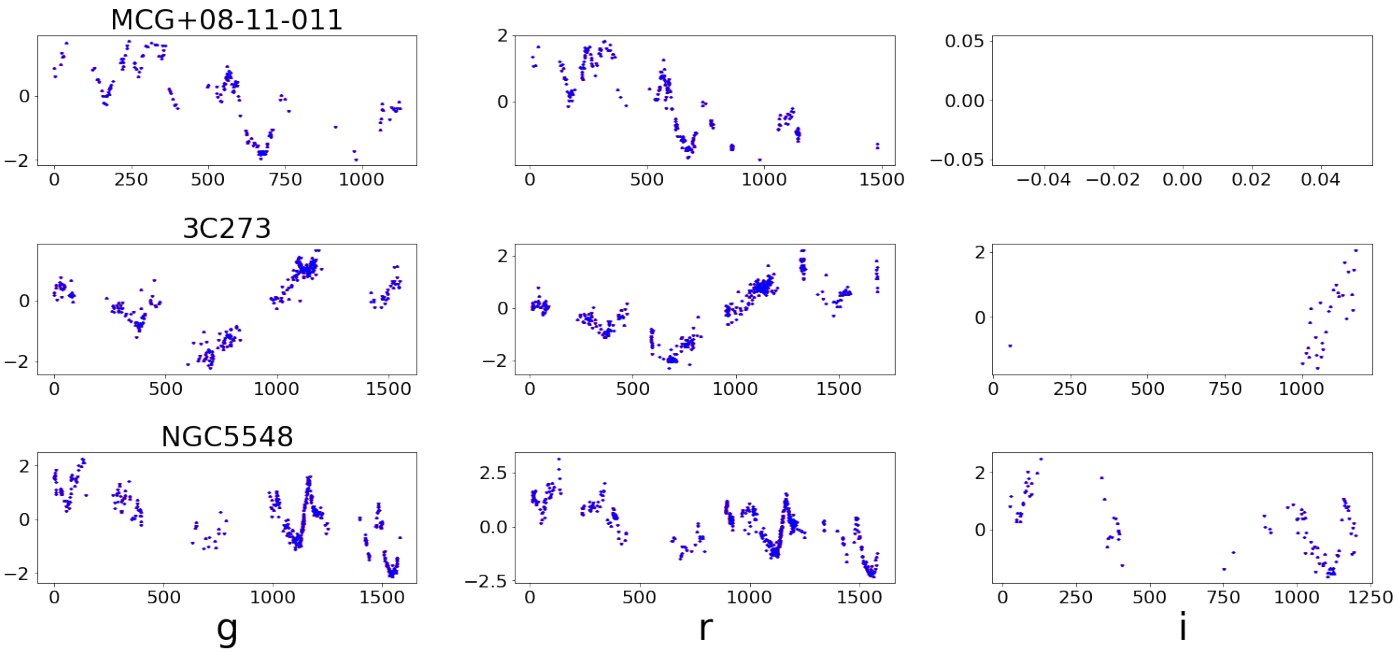


Figure 3.11: Normalized ZTF light curves for which objects we have known lags.

mechanisms used in the papers with the published lag estimates for consistency's sake and because cross-correlation analyses rely on the stationarity assumption and therefore the trends can bias the estimated lags ([70], [71]).

3.3 Summary

In this chapter we have discussed HeTVAE as a generative attention-only deep learning architecture with the functionality of projecting an irregularly sequenced, multivariate time series onto an arbitrarily chosen set of time points by *querying* the latent state. It builds off of the authors' predecessor, mTAN, by introducing a novel *intensity* attention mechanism to reflect observational sparsity in time and a heteroscedastic output

Object	Filt	$\overline{mag}$	N_{obs}	length	Median cadence	$\sigma_{nxs} * 100$
MCG+08-11-011	g	14.932	170	1122.008	2.961	0.041
	r	14.128	245	1471.008	1.914	0.032
	i	–	–	–	–	–
3C273	g	13.103	288	1550.961	1.965	0.003
	r	13.117	369	1672.270	1.848	0.003
	i	12.766	29	1120.0	5.471	0.000
NGC 5548	g	14.153	403	1579.848	0.996	0.023
	r	13.630	469	1576.895	0.988	0.011
	i	13.697	99	1172.855	3.971	0.009

Table 3.4: Light curve properties for reverberation mapping samples once the data is cleaned.

distribution that is able to represent this uncertainty in its predictions. It does so in a different vein than Gaussian Process Regression methods, which reflect variable uncertainty through posterior inference at the expense of significantly higher run times in both training and inference.

The forward process for the network is as follows: (1) embed both the reference time points and the light curve's observed time points to a higher dimensional plane. (2) for the *value* computation, at each reference time point embedding, take attention scores with all of the observed time embeddings using a compatibility function, softmax these scores and scale them by them by the magnitude values of the light curve before summing them together to garner a representation of magnitude at that reference point. (3) for the *intensity* encoding, the compatibility function is used between a reference time point and the observational time points of a particular light curve example as well as between the same reference time point and all of our *union_tp* points, where each set of calculations are summed and then divided. (4) concatenate the two attention metrics, and mix them linearly. (5) because the same embedding function and compatibility function is used for all dimensions, follow steps $1-4$ for each dimension of the data and after the linear mixing sum cross all dimensions. (6) for a new attention head and thus a new embedding function and a new compatibility function, follow steps $1-5$, then sum the representation across all attention heads. Steps 5 and 6 are encapsulated in equation 3.7. (7) is where

things change if we are using a true VAE or the added parallel pathway. For the true VAE, pass the encoded representation into an MLP creating two vectors for μ and σ, and sample. For the added parallel pathway, also pass the encoded representation through another MLP adjacent to the first, and concatenate it with the sample. (8) Now we can query this final embedding at arbitrary time points, performing the same computation as in the *value* encoding, and thereafter, (9) we pass this through a final MLP to produce a predictive distribution representative of magnitude at each of the time points we chose. If in the training phase (10), we would have originally used a subsample of the light curves in the encoding process and chosen the arbitrary time points to be the unseen time points, computed the composite loss 3.1.5 at such points and backpropagated.

As for our dataset, we have added a convenient preprocessing class to morph the dataset into neural-network compatible form, have allowed the ability to backpropagate from an augmented loss function that incorporates observational error and allowed for missing light curves to not disrupt the computations. We have also gone over the handful of HeTVAE-specific hyperparameters that were adjusted to make the network perform well on our dataset and hopefully are robust to other similar light curve datasets too. Moreover, we have made interfacing with the training configuration convenient and accessible via a legible *yaml* file and the command-line and have provided the ability to load and use pretrained models to make predictions. Finally, we introduced our methodologies towards the experiments we performed with results detailed in the following chapter. We briefly note that in all instances we are training and using the model with the parallel pathway 3.1.4 as was optimal for the authors. We could in the future remove this for the true VAE, which would probably be more pertinent to our filtering and anomaly detection experiments, but save that for future work.

CHAPTER 4

Results & Discussions

4.1 Training

We optimize for the objective using the Adam optimizer [75] with the standard parameter values ($\beta 1 = 0.9$ and $\beta 2 = 0.999$). Initially, we manually decreased the learning rate whenever the network returned a *nan* loss, starting from 1×10^{-4} to 3×10^{-4} and ending at about 1×10^{-6}. Stark jumps in the loss occurred when we were manually adjusting the learning rate between model checkpoints. The entire process was trial and error with countless hiccups in the process; for example, we accidentally reset the KL coefficient in code with each training pause and only noticed upon more closely monitoring and keeping track of the KL loss. Although only one example, this highlights the rather delicate nature of training these more complex models; easy to fix, but catching subtle bugs while training the model on hpc clusters is time consuming.

We concluded that it would be significantly more efficient, in that we would not have to supervise/manually pause and resume training on the compute clusters constantly, to implement a learning rate scheduler. We generally had to reduce the learning rate by a fraction when the loss stagnated during these manual adjustments, which made it

quite clear that an effective scheduler would be a decaying one. We chose PyTorch's *ReduceLROnPlateau*, which has several hyperparameters: the *threshold*, which sets how much the loss must change after *patience* number of epochs to reduce the learning rate by multiplying it with the $factor$. Observing the loss values and plateaus from the manually adjusted experiments led us to set the threshold to 0.01 because the plateaus occurred at that magnitude, a factor of 0.9 to smoothly decay the learning rate and a patience of ~ 100 epochs so as not to rush its reduction.

The KL schedule that seemed to be optimal was a bit surprising in that we only increased β, the kl coefficient, from between 0 and 0.1 to 0.2 for the single dimension models else the KL loss would totally vanish. We found that this *stop* value increased with increasing dimensions so that we used $0.3 - 0.6$ for the 2d models and $0.6 - 0.8$ for the 3d models. Otherwise we concluded with about $4 - 8$ cycles for every 1500 iterations with the time of β to be increasing versus flat β equal. With this schedule, it is clear that the latent space is compressing and decompressing to learn better structure, considering that we can see peaks and troughs in the KL loss for each cycle or equivalently when β increases in the cycle, the KL loss decreases and jumps when β is reset back to 0 (figure 4.1a). We briefly also note that the learning rate's starting value was able to be higher for the one-dimensional models (3×10^{-4}), but 1×10^{-4} for higher dimensional models even though we used 1×10^{-4} in all instances out of convenience.

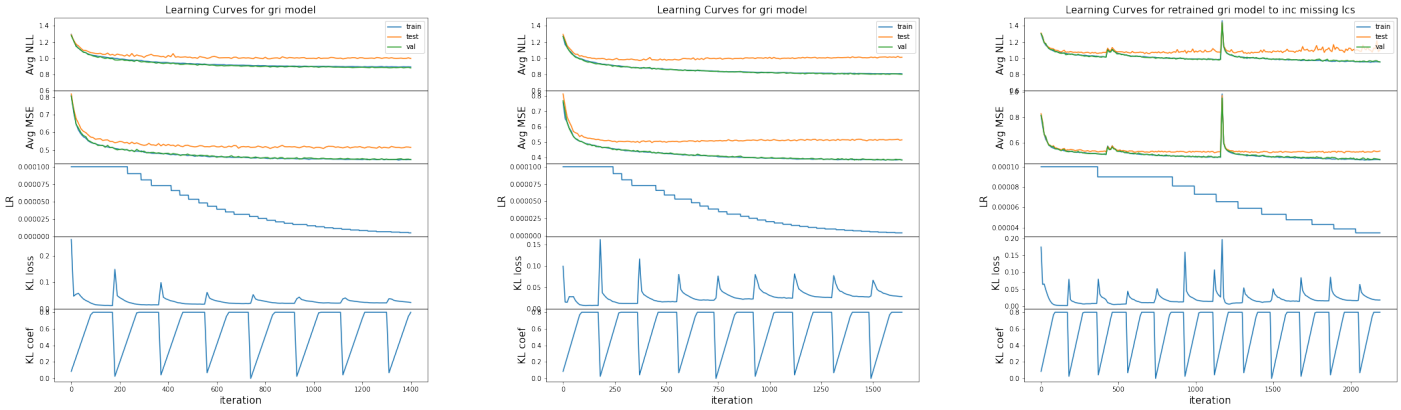
Taking a further look at the learning curves in the *gri* model, we can see that the test loss is stable, so that the size of the test set we chose as 10% of the training set is reasonable. The validation loss completely aligned with the training loss, which made us skeptical that it should likely be a separate group of distinct examples (instead of uniquely subsampled examples from the training set as HeTVAE's authors employed) in the future. It could be a potential bias against generalization performance because of its use in tuning hyperparameters.

Most models we trained converged at average NLL values of 0.8 to 0.9 and average MSE

values of 0.35 to 0.4. We ended up using only 8 attention heads in some instances because of how much slower it was to train models with 16, which altered the MSE loss on the order of about ~ 0.1. The MSE values seem quite high until one remembers that the model is predicting to points it has no information from in the encoding phase, as opposed to previous AE models that encoded the entire light curve and then took losses at all the points. For reference we present some of the interpolations of arbitrary g band light curves interpolated to a low resolution of 100 points between the light curve's ranges via the *gri* model as training progresses in 4.2.

There should be no harm training the network to reach as minimal a MSE and NLL as possible on the training set so long as the test loss does not increase drastically. In future experiments it would be more apt to apply apply k-fold cross validation, but we leave this for subsequent experiments due the sake of time constraint. We did not significantly experiment with dropout, but using a value of 0.1 seemed to reduced the gap between test loss and training/validation loss slightly, which is what we administered. For reference, the mean time it took to complete 10 iterations of training (with 16 attention heads and the *gri* model) was $\sim$ 2000 seconds (among 10 measured sets of 10 iterations) or $\sim$ 30 minutes. Extrapolating this to 1000 iterations is 60 hours on the GPU (a NVIDIA TITAN with 12.288 gigabytes of memory) without pauses in between.

We mention the hyperparameters were generally decided upon by the performance of the g model as it was very early on in the work, and we did not even know if training on the multivariate datasets would even pan out. Yet it was established later on that the main configuration parameter worth adjusting among increasing dimensions was n_union_tp, i.e. it was set to a uniform sequence of 3500 points between the ranges of the light curves, and changing this number to 20K for the *gri* model improved its NLL performance from ~ 0.9 to ~ 0.8. We suppose it was then less bottlenecked from being able to project enough information on the time series' sparsity. We show the learning curves with this increased n_union_tp in figure 4.1b. Finally, to show the model's capability to both be fine-tuned and to ingest uglier datasets, we show the learning curves of a *gri* model that



(a) (b) (c)

Figure 4.1: Learning curves for the *gri* model, including the average negative log-likelihood, the average mean squared error, the learning rate, the Kullback–Leibler divergence and its coefficient to reflect the cyclical annealing schedule. (b) is reflective of the *gri* model with an increased *n_union_tp* array as compared to (a). (c) shows the result of retraining an iteration of the *gri* model to include the examples that were filtered out because of not having a minimum length of 25 observations across all light curves. Its visible in (c) by the learning rate plot that the patience parameter in the *ReduceLROnPlateau* schedule was larger as it did not get decremented as quickly as in (a) and (b). All of these plots reflect the extent to which hyperparameters can be toyed with, (for better or worse) and are meant to give an example of how the learning curves look more generally across the other sorts of models were trained (be it the *g* model, *r* model, *gr* model, etc.)

was originally trained with examples in which the minimum length requirement for all the light curves was 25 and was subsequently retrained to include all missing light curves without a minimum length requirement (resetting the LR) in figure 4.1c.

We found that the model's performance in using it locally after training on the HPC to be inconsistent. It would return abhorrent NLL scores locally, but reasonably good MSE scores, which made it not so obviously noticeable until viewing the interpolations. As such, all the experiments,originally coded on easy to manage local jupyter notebooks, had to be written as scripts for the HPC system. We initially thought it could be a bug with the random seeds, but ruled that out, making sure the software packages controlling such things (PyTorch, Numpy, Pandas) were aligned and the same version, to no avail. Regardless, the notebooks are entirely viewable at `https://github.com/mwl10/hetast/tree/master/src/notebooks`.

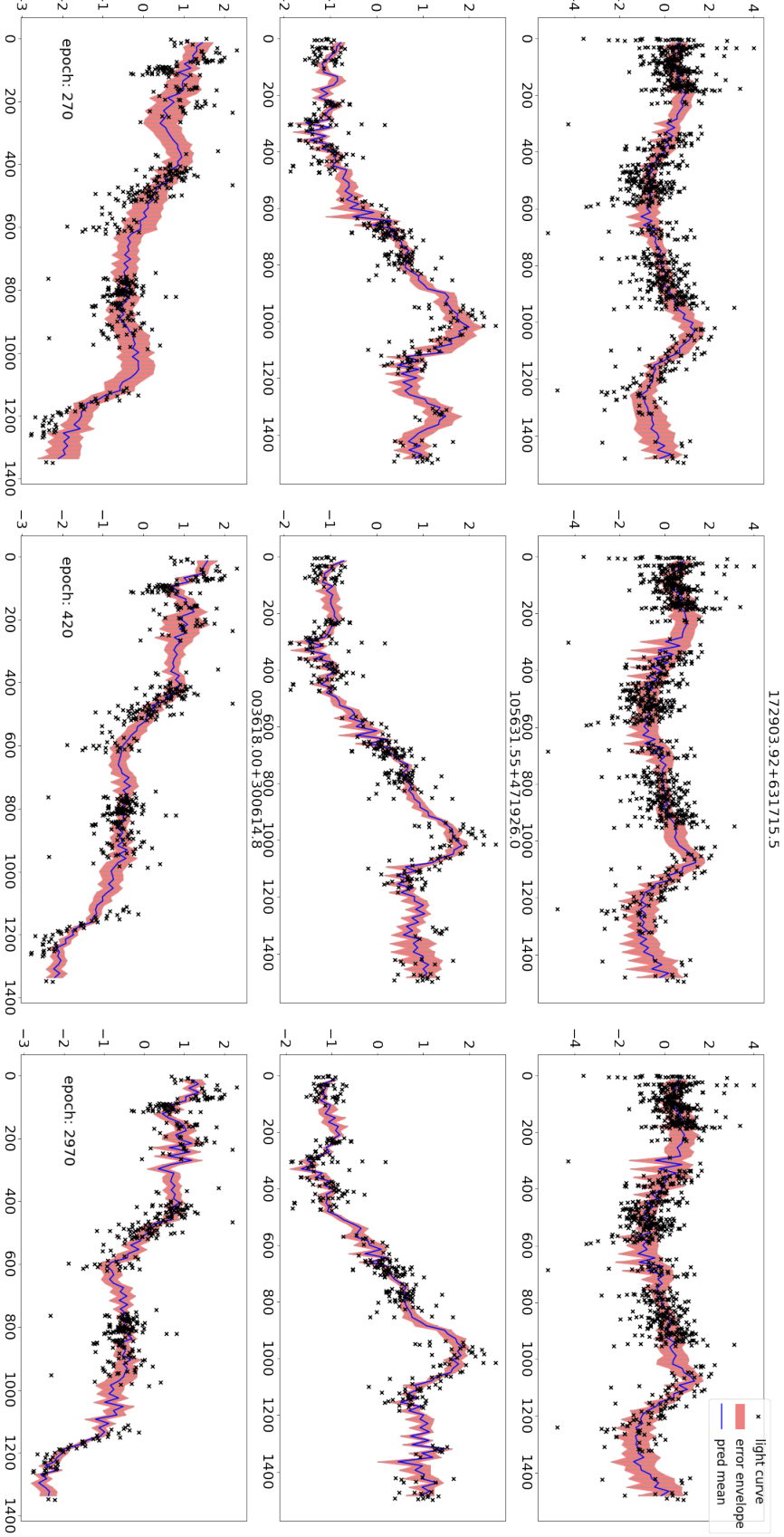


Figure 4.2: Example interpolations as training progresses on the *gri* model (but showing only the g band light curves) of three arbitrary objects with 100 points evenly spread between each light curve's ranges. This is before we implemented the learning rate scheduler, which shortened the number of iterations required to reach convergence. We note that the only change required to forecast instead of interpolate would be to change the array of target time points we choose to project to to the future versus between the range of the light curves.

4.1.1 Discussion

The process of coming to have a workable model was a difficult one. We retrospectively conclude the amount of configurable hyperparameters the model has as a weakness, and it forced us to turn to use more powerful tools like Hydra and Optuna to organize and experiment with them. It was doubly difficult having to bounce everything between the HPC system and locally. Nonetheless, some of the conclusions we drew were: (1) The KL annealing schedule stop parameter increased as we used more color filters/dimensions, likely because more dimensions made the latent space harder to contract. Moreover, upon looking at the number of KL cycles relative to the learning curves 4.1, this parameter could also be reduced, even halved, as the (visually) large number of cycles used might be interfering or distracting with the model's learning. (2) The loss landscape of the model was tricky and more delicate across increasing dataset dimensions so accordingly the learning rate was able to begin higher for 1d models (3×10^{-4}) than for 3d ones (1×10^{-4}) although we did not have to change the learning rate to be any smaller for datasets that were not filtered with a minimum length requirement. Almost always the loss would plateau for a significant period of time before suddenly dropping, and thus a larger patience value for the scheduler could mean that the network would be able to reach these random dropping off points in the loss, with the caveat that if set to be too long the network could return *nan* loss.

(3) Overfitting was not so drastic of a problem, and the test set size we had was acceptable, but a separate validation set would be more conservative and probably should have been used to tune hyperparameters. (4) We could stop and start the network training without problems, retrain a model to specific examples, a different dataset, for instance one less filtered 4.1c, functionally. (5) n_union_tp needed to increase across dimensions, and we have yet to determine some formulaic method for finding the optimal value of n_union_tp across different dataset varieties, but in the future could look at the intensity embedding scores to make sure that they are not too small and in between 0 and 1 so the sparsity information is optimally passed across (apart from just looking at the loss values to configure it). To find a formulaic best value for n_union_tp, we could

compare its optimal value to things like the cadences found in the dataset or the number of observations per light curve to sus out a pattern. (6) We note that we have yet to draw any conclusions about how the performance of the network scales beyond the size of our dataset, which at its largest, the *gri* dataset with no minimum length requirement, consisted of 3177 examples and a total of 8823 light curves across each of the dimensions (with at least 1 observation).

4.2 Using the VAE

4.2.1 Visualizing Attention

One of the advantages of using attention-based embeddings is that they should be somewhat interpretable. For instance, in HetVAE we can look at the normalized attention scores for the observed points with any one of the reference time points. Also, we have 16 attention heads for our model so we have 16 different ways that a reference time point can 'pay attention' to the observed time points in a light curve. Figure 4.3 shows a visual depiction of this, which highlights how the different attention heads have learned to look at different segments of the light curve, and thus different variability timescales, and in some cases, learned to look more broadly at the entirety of the light curve when discerning relevant summary information for the embedding.

4.2.2 Blending Light Curves

To show off some of the capabilities of the model being a VAE and to entertain our curiousity a bit, we can do several things. For instance, we could encode two arbitrary light curves, average their encodings, sample the average and decode the sample to create a 'blended' form of them. More specifically, we encode each light curve, average their discrete paths, average the mean and standard deviation output from the encoder, take ten samples then decode each separately with the target timepoints as the union of time points for both light curves, and finally average the separately decoded sample's interpolations. We choose two pairs of arbitrary light curves from our dataset and use the trained g model, shown in figure 4.4.

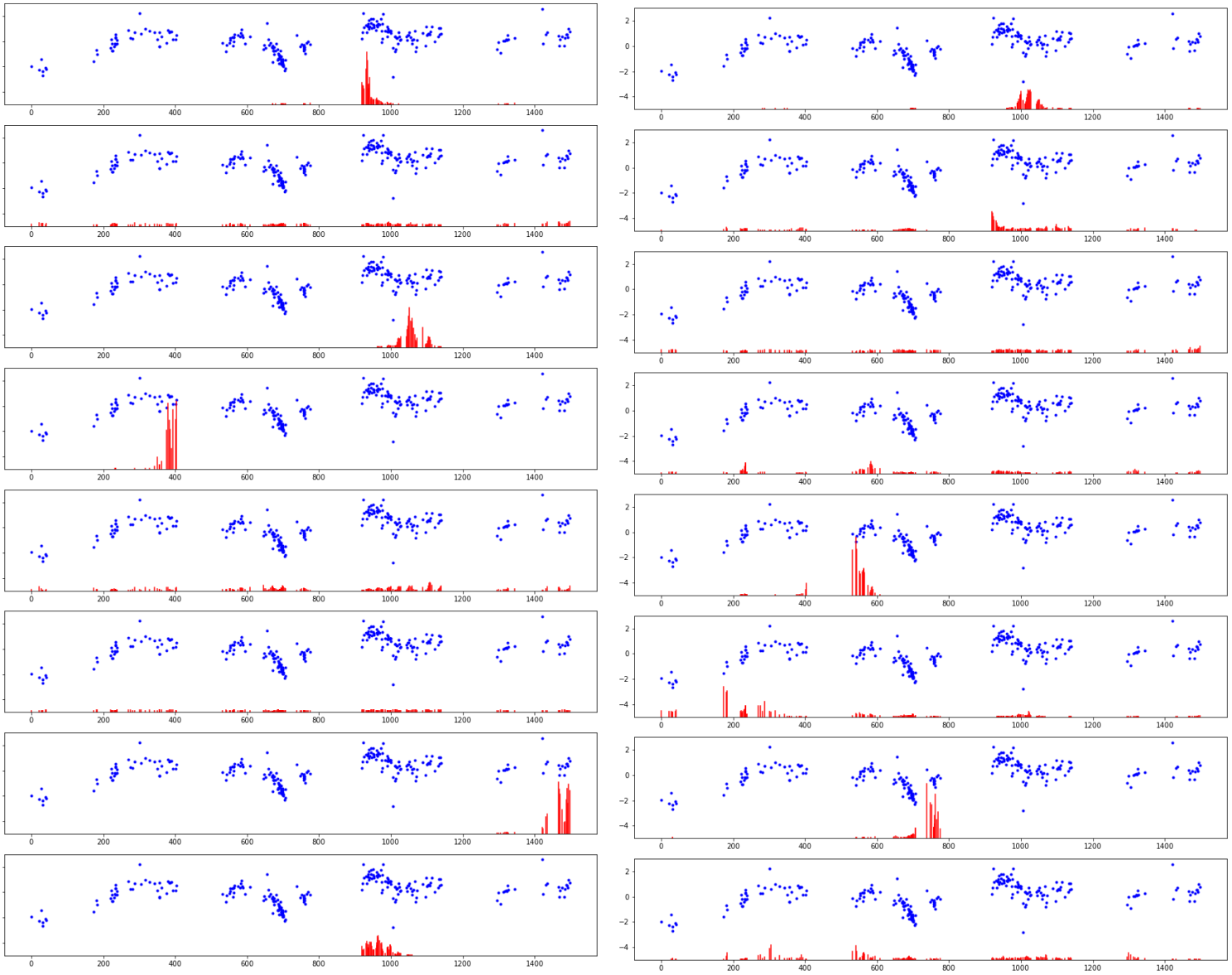


Figure 4.3: Visualization of the attention probability scores for the g band light curve of SDSS 080306.81+195953.1 for the univariate g model. This shows how much attention a reference time point (the first) is paying to the each of the observations in the light curve, which is different for each of the 16 attention heads. This is akin to the plot in 2.5 that shows attention scores via opaqueness.

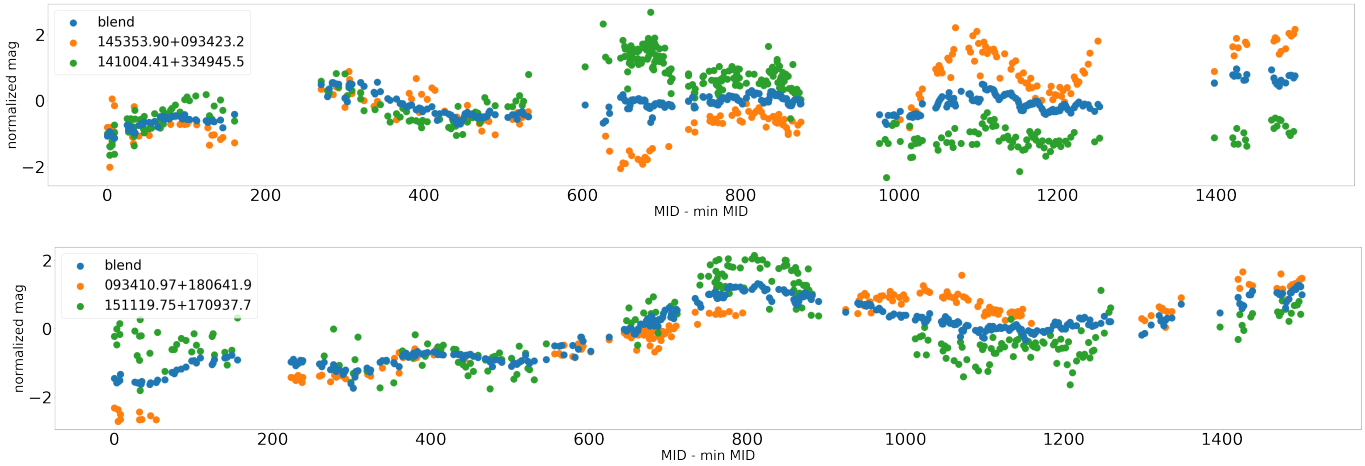


Figure 4.4: The blends of SDSS 145353.90 + 093423.2 and 141004.41 + 334945.5 and 093410.97 + 180641.9 and 151119.75 + 170937.7 with the g model. The x axis is in days, starting at zero for each light curve.

4.2.3 Generating Light Curves

Moreover, we can test the data generating capabilities of the VAE by sampling randomly from the latent space. In this case, we take a random sample from a Gaussian centered at 0 with variance of 1 of the dimensions num_ref_points (16) by $latent\_dim$ (64) and decode it with the network (using this sample in the discrete pathway too). We project the simulated light curves arbitrarily onto the cadence of exisiting light curves in our dataset and show simulated examples for both the g and gr models, as shown the figures 4.5 and 4.6 respectively.

4.3 Filtering Algorithm

4.3.1 PCA Binning

In this experiment, we use PCA to help us discern how light curves with different sorts of variability are distributed in the latent space. To see what variances in the latent space the principal components are pointing out, which should point out intuited 'differences' in variability of the light curves, we can bin the light curve's latent space embeddings after they are projected onto some of these components, average the embeddings within

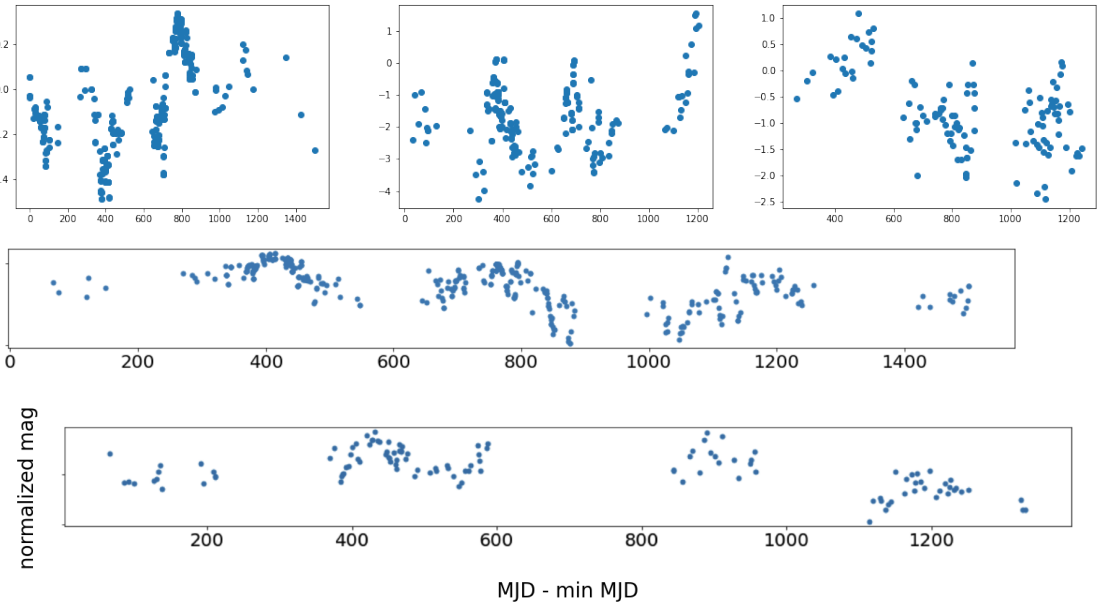


Figure 4.5: Simulated light curves with observational times taken from different existing light curves. On the y-axis is normalized magnitude, and on the x-axis is days starting at 0.

the bins and reconstruct those averages to see what the reconstruction looks like. We apply the same technique as in the blending section 4.2.1 here, reiterating, that is to average the discrete pathways for all of the light curves in a given bin, average their mean and standard deviations from the encoder and take 10 samples separately passing each sample along with the averaged discrete path to the decoder to make 10 reconstructions, which are then averaged. Whether or not we failed to mention this, the light curve is not subsampled before encoding now that it is not in the training phase.

We show a couple of different binning configurations using the g model: Namely, figure 4.7 makes six one-dimensional bins for each of the first 10 principal components separately, and 4.8 uses the first two PCA components to project the embeddings on a two-dimensional plane so that the bins are squares within the plane. For 4.7, we show the reconstructions projected on the time points of an an actual light curve (4.7a) as well as an unrealistic (4.7b) uniformly spaced cadence between 0 and 1500 days. Throughout these examples the majority of the light curves from the actual dataset 'look' like the middle bins, while the minorities are in the outskirt bins. We support this with figure

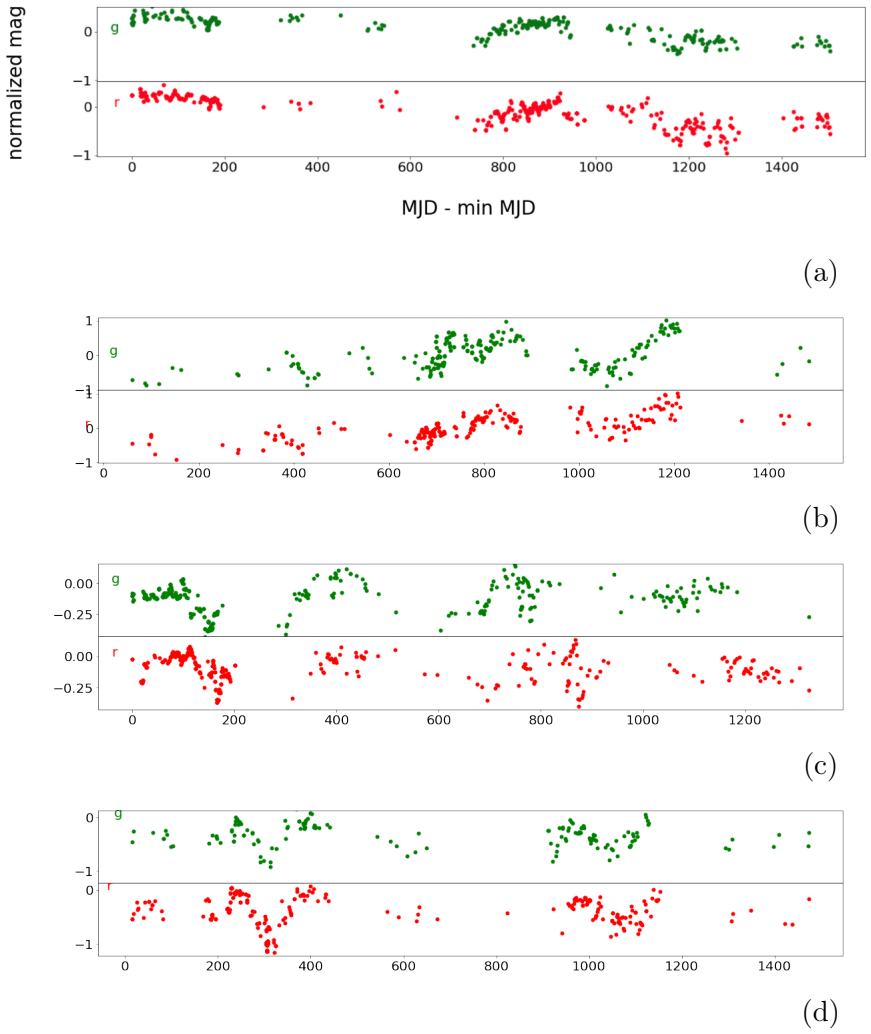


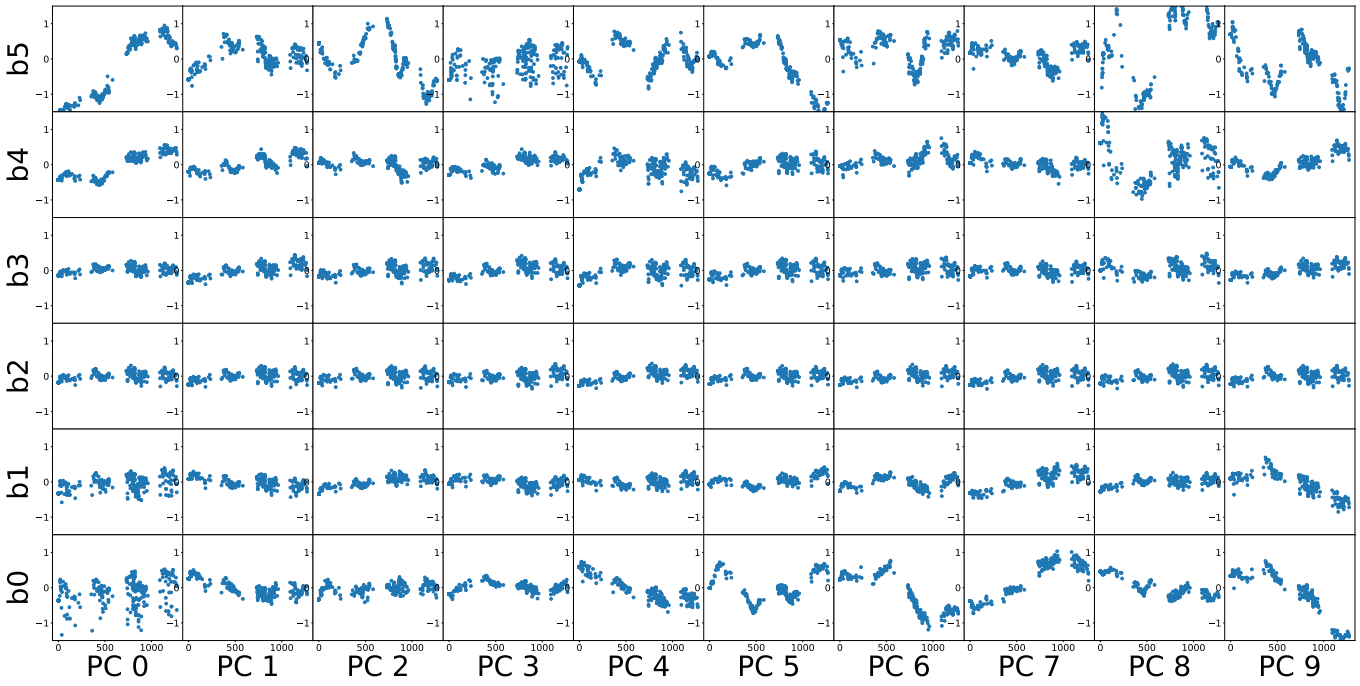
Figure 4.6: Simulated light curves after randomly sampling from the latent space of the *gr* model, projected onto observation times from randomly chosen light curves in the actual dataset. On the y-axis is normalized magnitude, and on the x-axis is days starting at 0.

4.9a, which shows how light curves from the dataset are distributed across the six bins of 4.7 for the first five the principal components. Considering that the latent space is trained to be a high-dimensional Gaussian, it makes sense that the way that the light curves are distributed in the pca dimensions matches this.

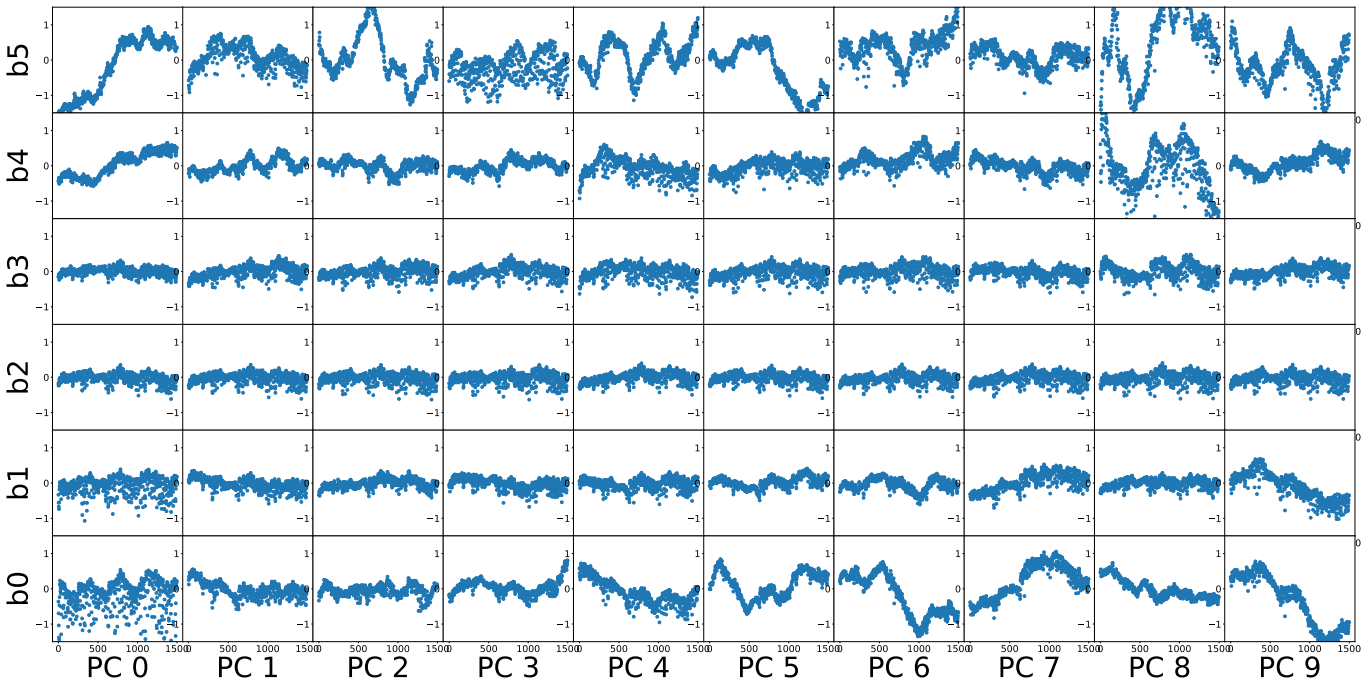
Additionally worth mentioning is that each principal component explains a percentage of the variance in the embeddings, sensibly entitled: explained variance ratios (figure 4.9b). The ratios for the first five principals components are $[0.0423, 0.02230.0205, 0.0170, 0.0108]$ and thus account for 11.297% of the total variance. The first 20 principal components account for 18.591%, the first 100, 33.679%, and the first $500, 78.289\%$. This is good because it leads us to believe that information content is somewhat reasonably spread out across the latent space features and importantly, that they are all relevant in some way for the model.

Now we hope that we can discern bins that we deem 'interesting' because in those bins are light curves that share such viewed characteristics of the reconstructions that we can further explore. But which are in fact interesting? Firstly, it is interesting to see how smoothly the light curves morph across bins (4.8, 4.7a, 4.7b), and this reflects the VAE's continuous latent distribution. Secondly, it is interesting to see the discrepancies made between light curves across bins. We notice that this gradient most often changes by way of the light curves with both low amplitude short-term variability and high amplitude long-term variability being placed in the top bins, light curves with little variability being placed in the middle bins, and light curves with both high-amplitude short-term variability and high-amplitude long-term variability being placed on the bottom bins. This discrepancy is very obvious in the bins that blend the first two components 4.8. To make the separation somewhat more blatant, we can calculate variability metrics for the bins like the sample variance, S^2, as labeled in figure 4.10.

In addition, we cannot forget that we can perform the same such pca binning experiments with the multivariate models so that in each bin we have a reconstruction for as many



(a)



(b)

Figure 4.7: Reconstructions of the averged embeddings in different bins for different principal components projected onto a realistic cadence from a light curve in our dataset (a), and the same reconstructions projected onto a uniform cadence (an observation every 2.5 days between 0 and 1500 days) to more clearly show the variability characteristics (b).

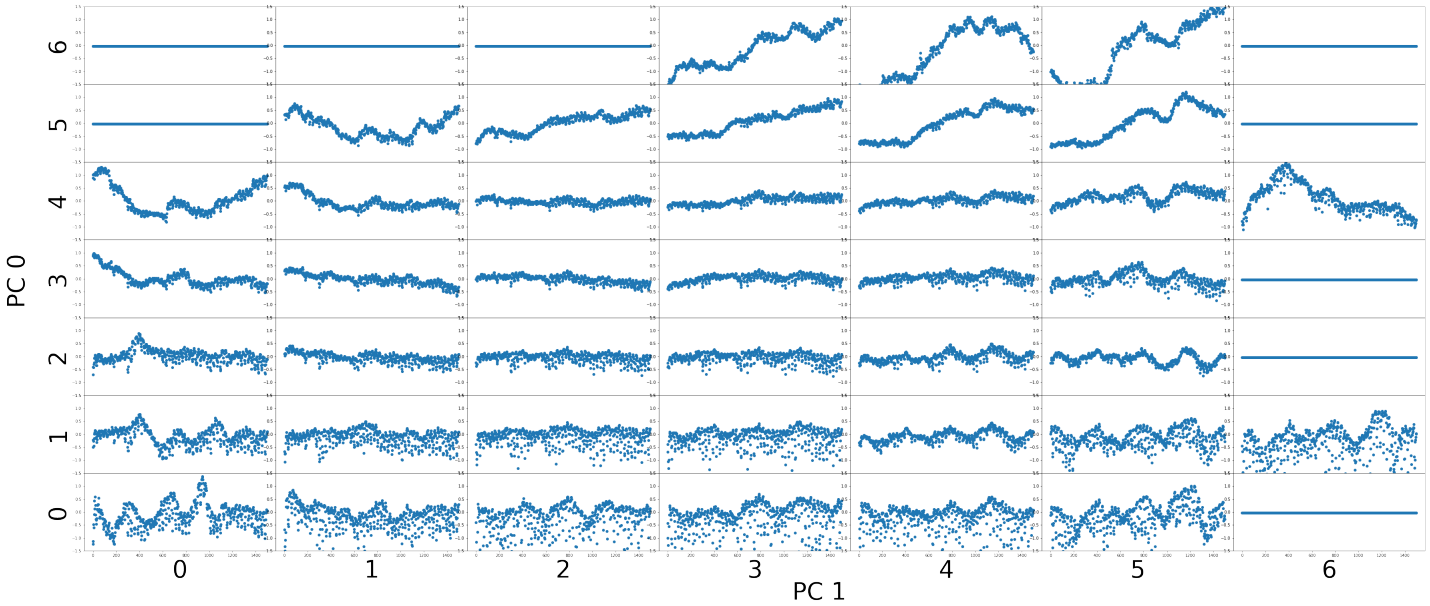
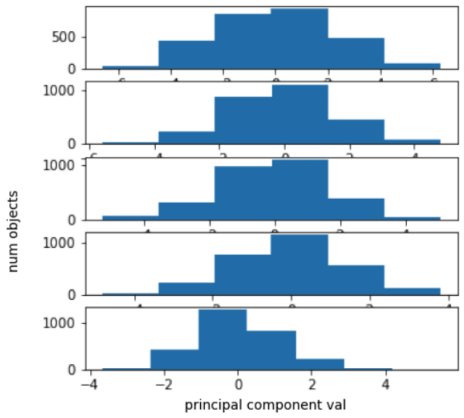
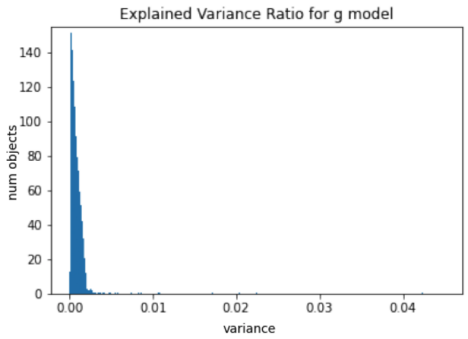


Figure 4.8: This figure shows the averaged light curve reconstructions from two-dimensional bins made by the first two principal components.



(a) Light Curve embeddings distributed along first five principal components



(b) Amount of variance from the embeddings explained by each principal component. A few outliers can be seen at 0.04 and several at 0.02 and 0.01, meaning these are the first few components that explain the most variance.

Figure 4.9

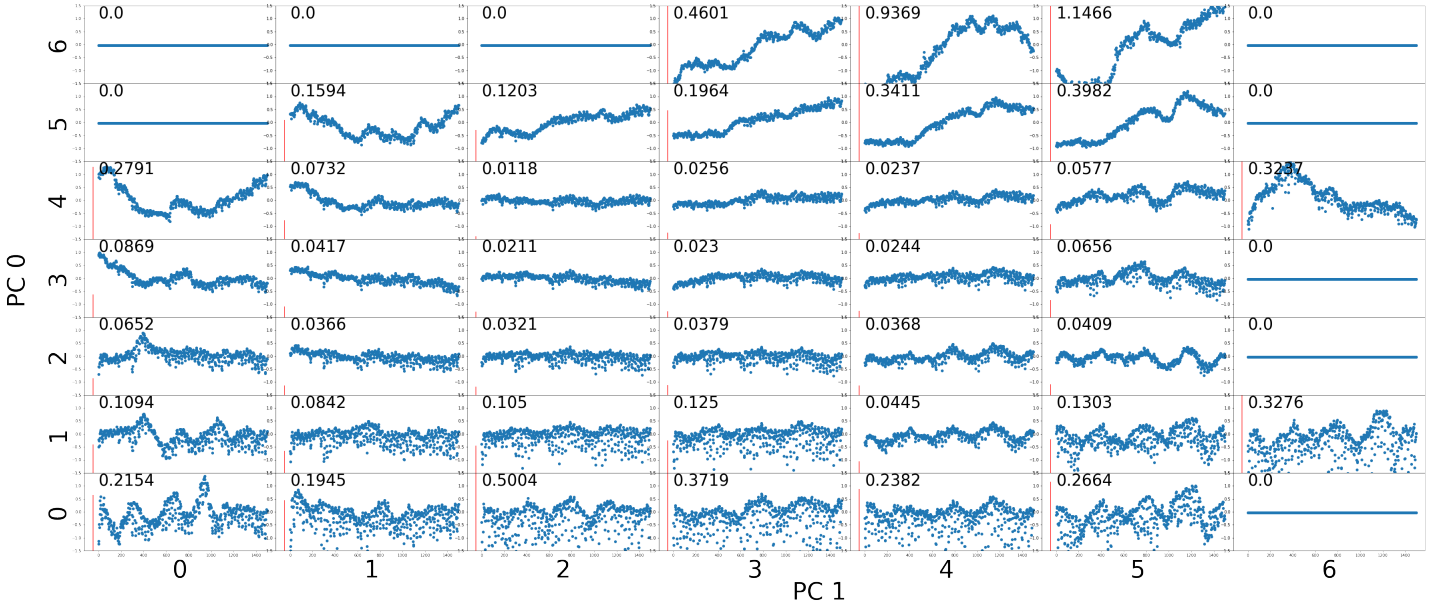


Figure 4.10: Showing the reconstructions from averaged light curves in the square bins created from PC 0 and PC 1, but with the sample variance labeled by the value (in the top left of all subplots) and a corresponding red line, the length of which is proportional to the value of S^2.

light curves as there are dimensions. In particular we show the analogous visuals of 4.8 and 4.7b with the *gr* model, making two-dimension bins with PC 0 and PC 1, figure 4.11b, while the one-dimensional bins for different PCs in figure 4.11a.

Now that we have shown some different binning configurations to reflect how the model learns to separate light curves in the latent space, we can choose bins and peer inside of them to reaffirm that the the reconstructions created are indeed indicative of the light curves placed inside of them. Consider that we choose bins because of their clear variability characteristics, making them great candidates for followup studies for a reverberation mapping analysis. Then any of the top row of figure 4.8 seem sensible to choose, (indexing by row first) that is the following bins: $(6,3),(6,4),(6,5)$. In the first of those bins, the reconstruction is the amalgamation of four light curves with SDSS names $142125.66+394328.8$, $083331.59+523405.2$, $225430.16+281655.4$ and $093410.97+180641.9$; in the second, $102734.60+073513.9$ and $114228.50+103229.3$ and in bin $(6,5)$ is $100621.69+124533.0$. In this case, we get more than one light curve for all but bin $(6,5)$ (if we wanted more in each of the bins we could just use less bins). Briefly glancing at the light curve in bin $(6,5)$ with the reconstruction and the light curve shown

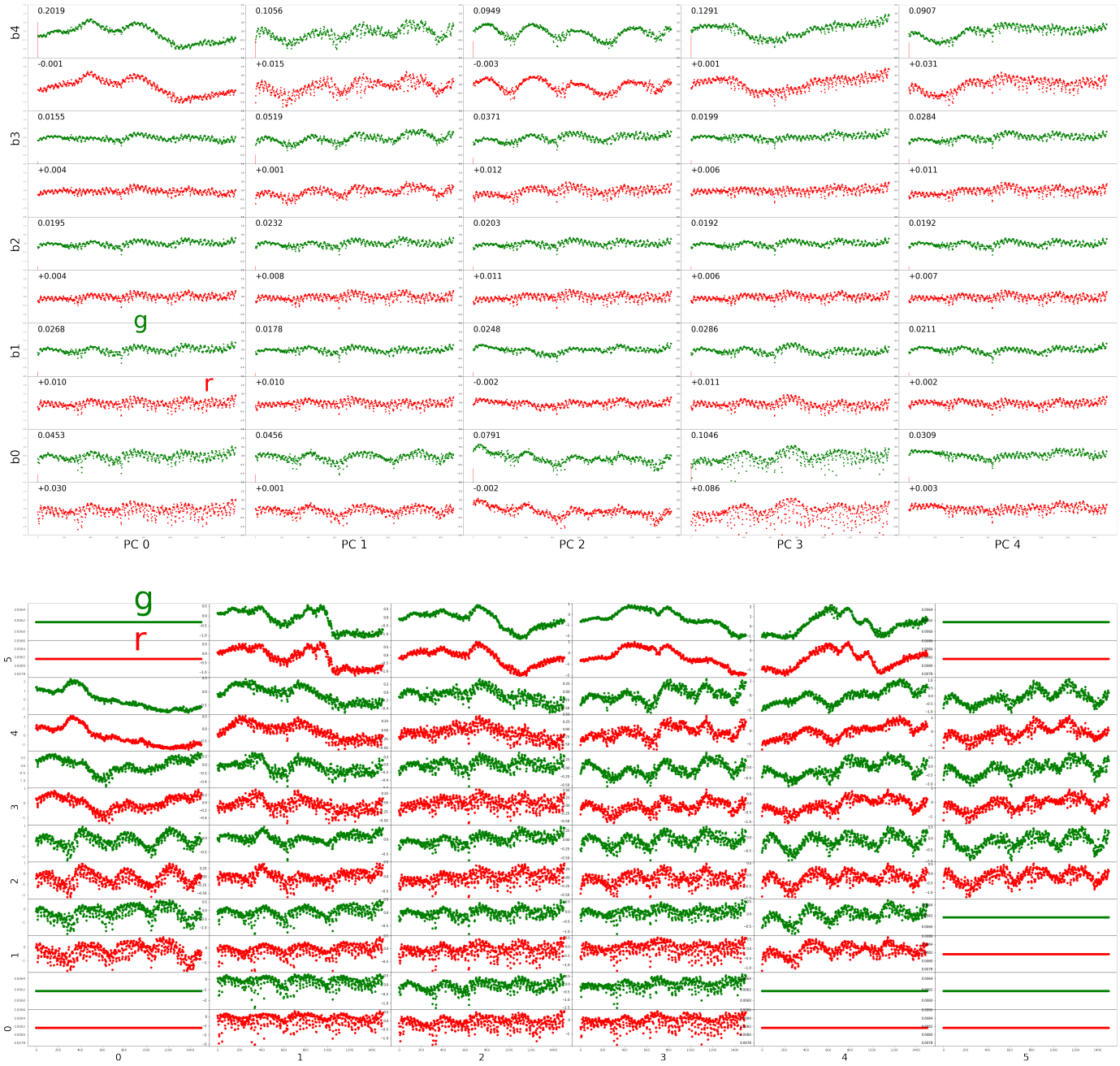
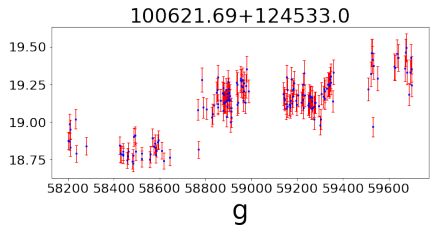
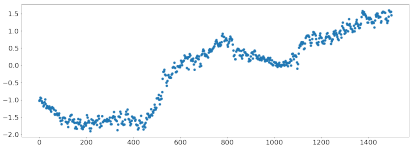


Figure 4.11: (a) is the the same figure as shown in 4.7b, but this time with the gr model. Specifically, the averaged light curve reconstructions from bins of different principal components, but this time we have a reconstruction in both the g and r bands for each bin. The red line shows the proportional sample variance, and for all of the r reconstructions, the relative sample variance to g is shown. In almost every instance, the variance for the r reconstruction is higher than its companion g; this is visibly clear as well, aligning with our understanding that variance increases with lessening wavelength. (b) is the same figure as in making 2d square bins with the first two principal components with the g model 4.8, but this time with the gr model so that there are reconstructions created for two light curves in each bin.

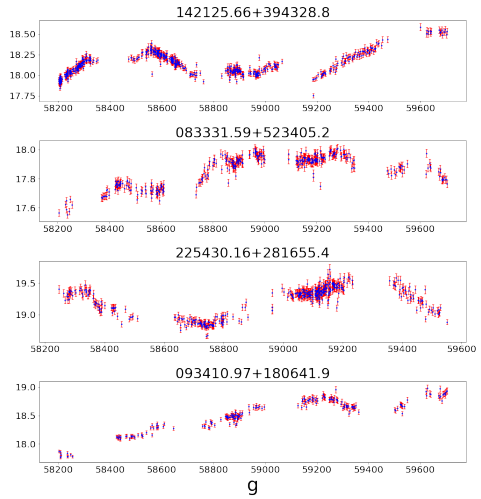


(a) Light curve that was placed in bin $(6, 5)$ of figure 4.8.

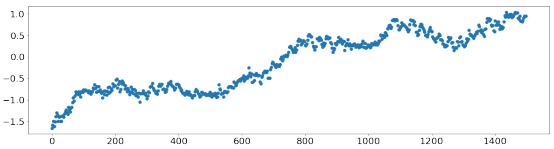


(b) The reconstruction from the light curve in the bin.

Figure 4.12



(a) g band light curves returned from bin $(6, 3)$ of figure 4.8 with SDSS ids labeled.



(b) The reconstruction from the light curves in the bin.

Figure 4.13

side by side in 4.12, we are somewhat reassured about not having made any clerical errors because the light curve looks identical to the reconstruction in the bin we took it out of (if one excuses the scaling and normalization distortions). We also show the light curves from bin $(6, 3)$ side by side with their reconstruction in figure 4.13, noticing that while all of these light curves somewhat show a globally increasing trend, they are most visually similar by way of their relatively similar magnitudes of fluctuation on different time scales.

The filtering aspect of this experiment is such that if we had more ZTF light curves we could encode them, map them on to the pre-made PCA dimensions and bins and

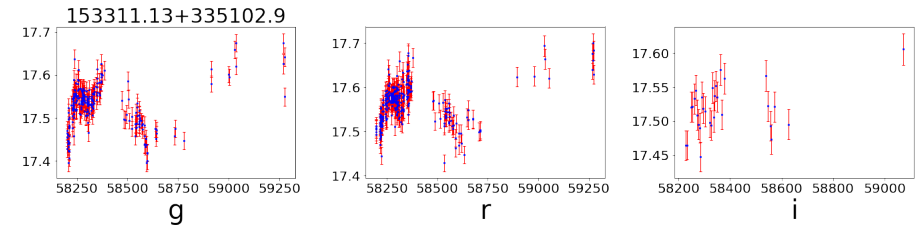


Figure 4.14: Object whose light curves end up in PC 8 bin 5 of 4.7.

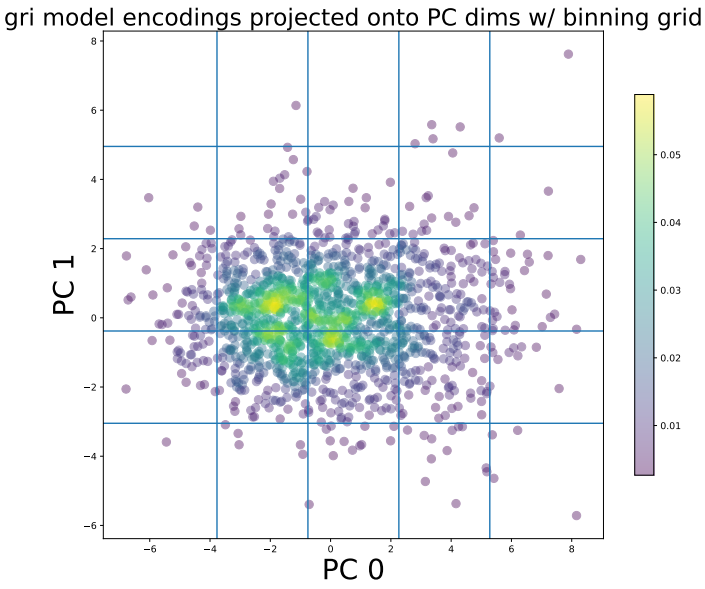


Figure 4.15: To clarify what is happening, we show the light curve embeddings, in this case from the *gri* model, projected onto the first two principal components and draw lines to show the bins separations. Whichever light curves embeddings appear in a given square are those whose embeddings are averaged and subsequently reconstructed to reflect the bin, i.e. these 2d bins are where the reconstructions come from for 4.8 and 4.11b.

remove/keep light curves according to which bins they end up in. Most likely we would not have much of an interest in any of the bins that reflect weak and structureless variability patterns, which is the vast majority of the dataset given that most often these types of light curves are in the middle bins. We could remove such bins or we could instead grab light curves out of very specific bins. For example, PC 8 bin 5 of 4.7, as the light curve(s) in this bin look to have some of the highest amplitude variability changes. In this case there is only one object, 153311.13 + 335102.9 shown in 4.14. We mention briefly that the encodings for $\sim$ 3k light curves take less than a minute to create locally, which makes this procedure quite fast to do.

4.3.2 Isolation Forest

We now use the zs from the gri model and search for anomalies using an isolation forest. The benefit of this method two-fold in that it takes into consideration all of the latent space features in tandem when deciding on anomalies and provides a score to each object so that we can dial-in a threshold wherein below it, objects are 'anomalies.' Thus we can also choose how many objects to focus on by this threshold. Yet these anomalies are vague and might highlight light curves that are anomalous due to parameters we already can calculate. We have seen in the binning experiments that light curves with particularly high variance will be anomalous because the dataset is mainly characterized by minimally variable light curves and thus the network is better adapted to them, and indeed variance, or some proxy of it, is a calculable trait so this bias is just a distraction from more candid anomalies. Catching this issue shows the benefit of doing the PCA experiments, but other biases would likely not be as visually obvious. Therefore, we will want to get an idea of what other parameters this model specifically might be sensitive to so in the future we can mitigate them, especially if we move to larger datasets later on.

The mild approach we take is to garner a list of objects that are anomalies with respect to calculable traits like black hole parameters (bolometric luminosity, redshift, black hole mass, eddington ratio) or other light curve characteristics like anomalous range, median cadence (sparsity), etc, and see which overlap with the IF's returned anomalies as well as look at their anomaly scores from the IF. The intersections that breed the largest amount of objects likely indicate which parameters the network is sensitive to and thus should be counteracted with fairness tactics in the future to prompt finding more interesting anomalies. We acknowledge that this method is somewhat hand-waving, but nonetheless might offer some insight as to what AGN parameters the model is most surprised by as per their affect on variability.

The authors of the RVAE used in [40] balanced their dataset with respect to black hole mass, the number of observations in the light curves and bolometric luminosity by capping

the amount of objects allowed in the dataset for a given range of the parameter values. In our model the number of observations is normalized out by the loss function, which is another way to mitigate a bias/dataset imbalance. Redshift seemed to be a disrupting parameter for their experiments, because of host galaxies intruding emission and/or because of time dilation and rest frame difference. Fortunately, with respect to z, this dataset is already about uniformly distributed between 0.6 and 0.65 and thus balanced as can be seen in 3.6, but we can look to see if the objects we added for reverberation mapping purposes show up as anomalous because their redshifts are lower as compared to the rest of the dataset; the highest being 0.158.

We employ the *gri* model trained only on objects whose light curves all have greater than 25 observations even though we have another retrained version of it (refer to figure 4.1c) to include the objects whose light curves did not meet this requirement. This is because we realized the retrained network would be biased to perform better on the light curves it was originally trained on, and thereby the latently added light curves would appear anomalous. In future experiments, we would just train a model to include all the missing light curves from scratch, especially because with the minimum length filter being 25, there are only 1527 objects to meet the requirement, while if it is 0, essentially all of the objects in the catalog (3177). Now extrapolating to massive datasets, imagine how much of the data distribution is lost if the filter is applied.

Moving forward to configure the isolation forest, we sample 10 times from the qZ encoding distribution and average those samples for each object. We apply 500 estimators for the forest, and gauge our threshold by the lookings of the returned anomaly score distribution, shown in 4.16. Below -0.46 yields 39 objects, while below -0.45 yields 135. We choose the latter as the IF list to compare with to err on the side of caution. In finding sources of bias, we look at the top 2% of anomalous black hole masses, bolometric luminosities and eddington ratios, either as larger or smaller than the mean, as the dataset is distributed relatively Gaussian-like with respect to these parameters (see 3.6).

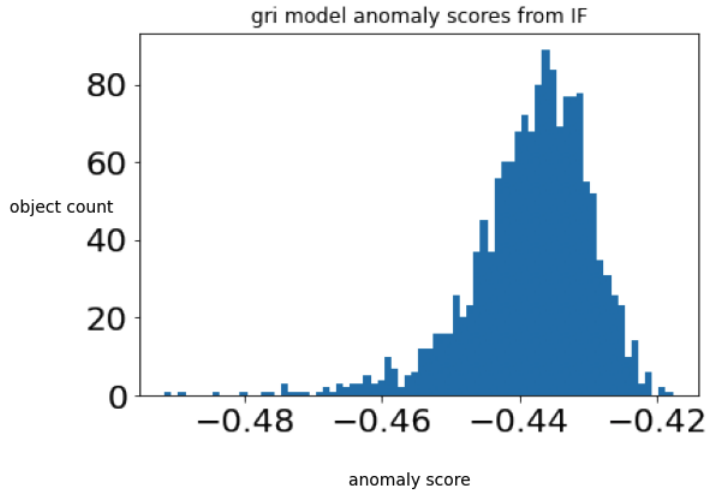


Figure 4.16: Anomaly scores from the encodings for the *gri* model with the minimum length light curve filter set to 25.

Of the 30 objects that constitute having the top 2% of outlier black hole masses, they are all greater than 2.05 standard deviations from the mean, and only two of them are in the greater IF list. The average anomaly score for these objects is -0.435, which is comfortably in the mass of the anomaly score distribution 4.16. The object with the most anomalous black hole mass (9.4) had an anomaly score of -0.428 and was 3.36 standard deviations above the mean of black hole masses.

Doing the same for objects in the top 2% for having anomalous bolometric luminosity, there is 1 mutual object with the IF's list, these objects are are all at least 2.365 standard deviations above the mean and they have an average anomaly score of -0.437. The object with the most anomalous luminosity of 46.745 $ergs\,s^{-1}$ had an anomaly score of -0.438 and was 3.72 stds above the mean. As for eddington ratio, 2 mutual objects with the top 2% that was 1.92 standard deviations above the mean, having an average anomaly score of -0.438. The top outlier was 2.85 stds above the mean with an anomaly score of -0.438.

Briefly looking at the anomaly scores of the objects we added to the dataset because their redshifts are particularly low, those that were not filtered out from this dataset being Mrk876 with redshift (0.1211), Mrk817 (0.1584), 3C273 (0.1583), NGC5548 (0.0163),

Mrk142 (0.0446), NGC2617 (0.0143), have the following anomaly scores: -0.435,-0.430,-0.440,-0.438,-0.433, which again is not so abnormal.

Now we peer at anomalous range and median cadence. We suspect that anomalous ranges could pose an issue for this network, as the way the reference points are distributed likely makes performance more optimal for the mass population, and we did not normalize the time points. The ranges of observations of the light curves sensibly align with the surveys time frame so that the g and r light curves are mainly 1500 days in length, while 1000 for the i. Now unlike the black hole parameters, the light curves are not normally distributed with respect to these parameters. There are not necessarily outliers that have uncharacteristically long ranges, but there are those with shorter than normal ranges (left-skewed). In this case we average the ranges across light curve dimensions and choose the objects within the 2nd percentile, which is 1001 days. The average anomaly score across these objects is -0.437. The shortest range was a very outlying 579 days. It had an anomaly score of -0.440, and thereby none of these objects were mutual with the IF's list. We acknowledge that this does not rule out the model's sensitivity to longer than average ranges, which we are just as if not more suspicious of although we leave this for future testing.

Objects whose light curves have unusual median cadences also could distract from the anomalies returned by HeTVAE, i.e. examples with extreme data sparsity. Once again, we average this value across dimensions. The distribution is significantly right skewed so we look above the 98th percentile or 4.98 days and get an average anomaly score of -0.437 for those objects. The object whose light curves had the greatest median cadence, a very outlying 8.7 days, had an anomaly score of -0.443.

Moving on from looking for sources of bias, we gaze through the returned anomalies from the IF list to look for promising CSAGN candidates because we know the model finds high-amplitude variability features anomalous and thus CSAGNs could be a subset of the returned anomalies from the model, as they are usually visually picked out by unusual

flaring activity or abrupt decreases or increases of observed magnitude/flux. We show five interesting objects from the top twenty returned list of anomalies with SDSS names $145950.60 + 065302.5$, $131800.84 + 092822.5$, $161344.89 + 283014.2$, $004406.58 + 082536.9$ and $211200.97 - 003635.2$. In particular, three of these objects ($161344.89 + 283014.2$, $131800.84 + 092822.5$ and $211200.97 - 003635.2$) show a 'plateau' feature as detailed in [40] characteristic of CSAGNs, where the object presents a spell of low amplitude variation or even constant emission before/after very abrupt variations.

4.3.3 Discussion

While the binning experiments pointed out impressive semantic information the VAE has extracted, they showed us that the model was quite biased by the excessive amount of light curves in the dataset with little to no variability. Thus the model's performance was more optimal for these light curves and any light curve with excessive variance was claimed anomalous. If that is satisfactory, and if we wanted to sift out more variable light curves, then we were successful. We were able to characterize the two different subgroups of variability we mentioned (high amplitude long-term and short-term versus high amplitude long-term and low amplitude short-term), but we had no way of differentiating between variable light curves otherwise because of this bias, meaning that the network was not even able to separate out light curves that would show anomalous variability due to having anomalous black hole parameters (given that none of the list intersections were significant), much less able to separate out the nuances of truthfully anomalous objects.

The main (and naive) mistake of ours was to not employ the dataset filters pre-training that simply removed light curves that showed little to no intrinsic variability (i.e. one of normalized excess variance/intrinsic variance or P_{var}; [40]). But more recent research into algorithmic fairness tactics use the VAEs themselves to point out systemic bias in datasets and adaptively resample the dataset based on the biases that the latent space points out. For example, in future experiments, we could first train a VAE as we have already to point out light curves with very little variability. Then we could remove them because we are interested in studying only variable light curves, i.e. by removing those in

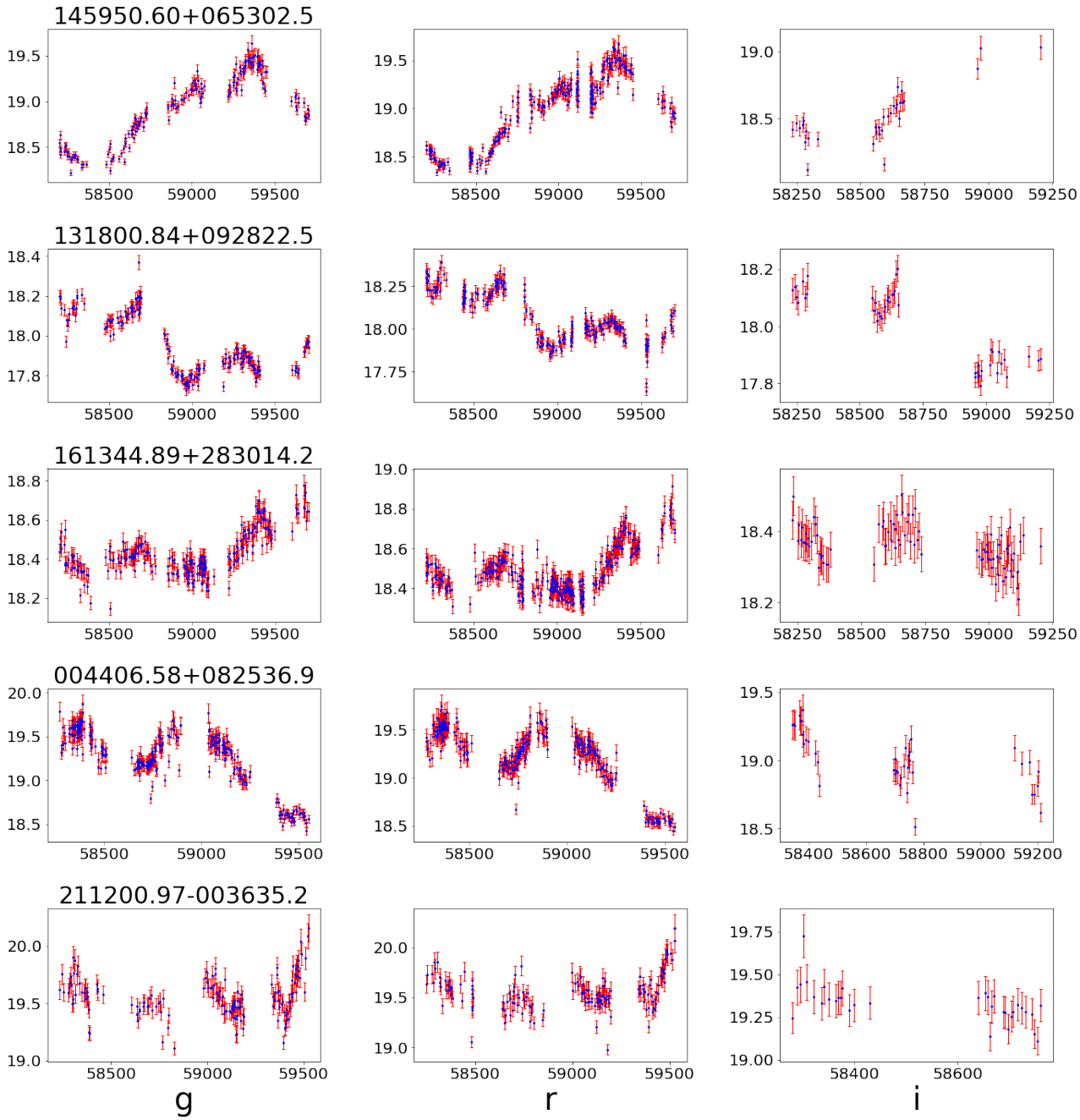


Figure 4.17: Five interesting objects from the top twenty anomalies sussed out by the *gri* model.

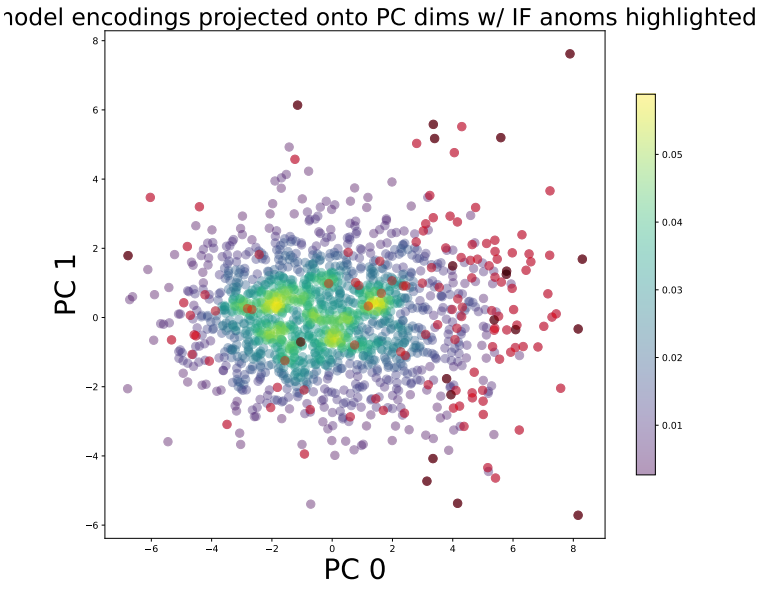


Figure 4.18: The light curve embeddings from the *gri* model projected onto PC 0 and PC 1 with the the 135 IF anomalies pointed out in red and the top 20 in black. Here it is clear that not all of the anomalies from IF are also anomalous w.r.t. the first two PCs as some are in the middle of this density plot.

the PCA bins that show little to no variability. Following this, we take our newly filtered dataset and train the VAE with it, focusing on mitigating bias with respect to parameters of the objects/light curves that we already know of to then reveal more candid anomalies.

We also briefly mention that we found the IF's returned anomalies were disrupted because of the fact that we had not yet with trained the model to include observational errors in the loss function so they would be punished just as much for not fitting points with high error bars as for with points where the opposite is true. A model trained via a loss function weighted by errors means that light curves with higher error bars would be easier to fit and thus be less anomalous.

On a more optimistic note, our results show that the network can harness and comprehend multicolor AGN light curves unlike previously applied deep learning algorithms. We are thereby eager to apply this model to a much larger dataset to test its scalability as we understand that realistically one could visually inspect a dataset of this size and that our

dataset is likely not entirely characteristic of the more general AGN population. Based on our results, we would also be remiss not to reiterate the necessity considering algorithmic fairness and of having an astronomer in the loop when applying machine learning to such a problem (and more generally to have a domain expert supervise the use of an ML model in any applied context).

4.4 Reverberation Mapping

In the following section, we move object by object, using PyROA on the light curves to get an idea of what lags can be estimated without our interpolations and then use the interpolations to see if any improvements can be made; this includes detailing the fine-tuning procedure we employed to get the best possible interpolations.

Based on our initial description of PyROA in 1.3.3, we have a few major toggles in the algorithm's usage. The first is the prior on the degree of blurring introduced by the size of the running average's Gaussian window Δ with the default being $0.01 - 10.0$ days. More blurring, or a higher window, implies less parameters, less close of a fitting to the curves and an improved ability for the ROA to stretch across cadence gaps. Moving forward we often use the same three increasing window size priors for the Markov chain: the default blurring of 0.01-10 days, a moderate blurring of 5-15 and a high blurring of 10-30 with initial values of 1,10,20 days respectively. The second toggle is the transfer function, which we change from the default dirac delta function to add another degree of blurring in estimating the lags for each band. We utilize the log-Gaussian transfer function, which, in addition to the extra blurring per light curve, imposes a more physically-inspired situation in that lags can not be negative.

All of the algorithm's other default priors are: $[0.0, 20.0]$ for the rms, $[0.0, 100.0]$ for the mean, $[-50, 50]$ for the lags and $[0.0, 10, 0]$ for the extra error blanket. The default number of walkers for each parameter is hard-coded to be $7N_l + 3$, where N_l is the number of light curves, and the default number of samples is 10k with 5k burn-in. For simplicity's sake, in every instance of using the algorithm henceforth we take 50k samples and remove

the first 45k to do our best to ensure the chain has stabilized and the samples are thus from the true posterior. In the following epoch separation visuals, we draw lines between the observational gaps based on them being a given number of standard deviations above the mean cadence, dt. The number of standard deviations is chosen for each light curve relative to what seems to most optimally separate the groupings of observations.

Now after we gather the best guess lags from the PyROA algorithm, which, if nothing else, will give us a bearing on how well we can estimate time-delays using relatively low cadence ZTF light curves, we move to use HeTVAE's interpolations of the light curves in conjunction with PyROA in the ultimate hope that we can align those lags more with the published estimates. We mainly use PyROA to appropriate its blurring transfer function and because it does not really inject any additional information otherwise as it uses a running average.

We found that the interpolations of the light curves when training on the entire dataset were underfit and jagged, but aligned to the general shape of the light curves well, i.e. figure 4.2. The model did not reach for extreme and unique variability features of individual light curves because that would reduce its general performance. In addition, the more clearly variable light curves were outliers so the performance on light curves of that variety, indeed those optimal reverberation mapping, is less than perfect for that reason. The vagueness of the interpolations is a problem for reverberation mapping analysis, where it is mostly only beneficial to have highly particular interpolations. Now these interpolations could be very specific to the individual light curve and not draw from a general concept of AGN variability, much like PyROA's running optimal average; in fact, we could have highly the same such individualized interpolations with HeTVAE if we trained with very small datasets or even individual light curves, which we will subsequently show.

But luckily the solution is somewhat straightforward with the models we have trained already, that have a solid generic understanding of AGN variability, and that is fine-tuning. The idea is to resume training from the best checkpoint with the light curves

we want better, more specific, interpolations for. In this case we have the best of both worlds such that the interpolations still draw from the general ideas of AGN variability that the deep learning model has intuited, but so too do we have the specificity required for individual light curves. We would be able to do this locally if not for the bug with models performing differently on our local computers after they have been trained on the compute nodes. If the issue was resolved, so too could the general user with access to the model checkpoint fine-tune the model to arbitrary ZTF light curves or potentially even more out-of-distribution (OOD) light curves (like those from other surveys) locally because it is not very computationally expensive to do so.

More specifically, when we fine-tune on the light curves, we turn off the kl-annealing schedule, while resetting the original *ReduceLROnPlateau* schedule for the optimizer to the same initial loss of $1e-4$, but with a patience of a much larger $\sim$500 epochs to squeeze out performance. In addition, we resample additional copies (10) of the light curves using the method of equation 3.17 to add information about their observational errors because we have not yet trained on the augmented loss to include them. When we make the interpolations, we follow the same steps as in making the reconstructions for the PCA bin experiments, but average the interpolations across the resamples of the light curves, and might as well choose as highly resolved of a cadence as possible to interpolate to, computationally permitting; this was a cadence of 0.1 or 0.2 days as otherwise we would get 'gpu out of memory' errors.

4.4.1 MCG+08-11-011

The previously published lags for MCG+08-11-011 were generated using Javelin in [72]. Javelin by default linearly detrends the light curves. We do the same in an attempt to remove as much bias as possible from our experiments, regardless of whether or not the light curves appear stationary. These reference lags were in the *ugriz* filters so we subtract the g lag out from r and i as it is our available driving light curve. The light curves used to generate them had an approximately daily cadence over 4 months in 2014.

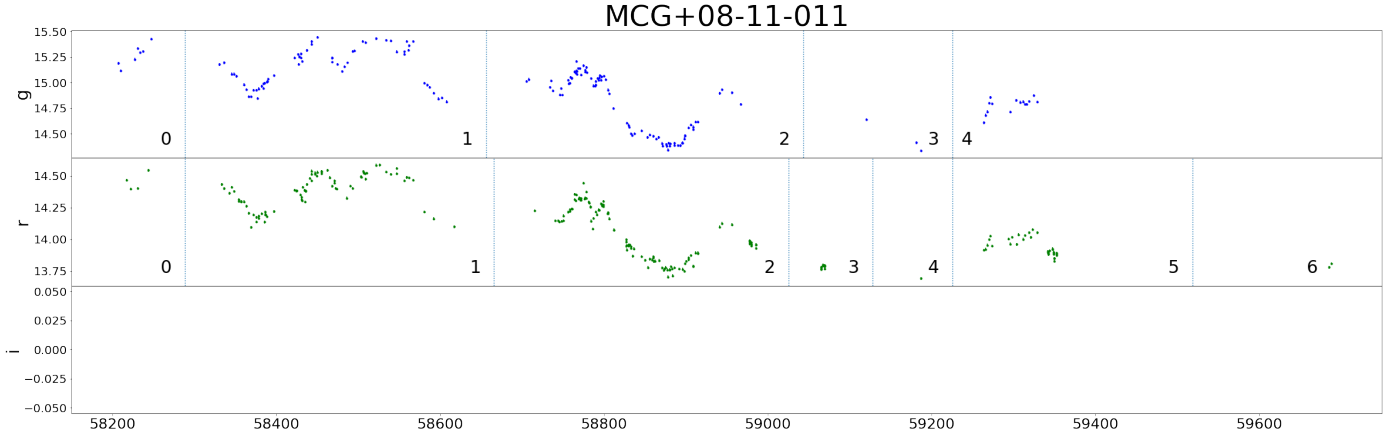


Figure 4.19: Epoch separations based on *dt* being a given number of standard deviations beyond the mean (whichever number of stds for each of the light curves so that their epochs would best align across bands).

For the MCG+08-11-011 ZTF light curves, with epoch separations shown in figure 4.19, we do not have a light curve for i, so we focus PyROA on the g and r epochs with noticeable variability features (epochs 1 and 2). These segments are both approximately 300 days in length (∼10 months); (1) from $\sim$ 58300 to 58600 MJD and (2) from $\sim$ 58700 to 59000 MJD with a median cadence of about 3 and 2 days for the g and r bands respectively. Thus in spite of the lower ZTF cadence, we have longer periods of observations to study and base lag estimates off of.

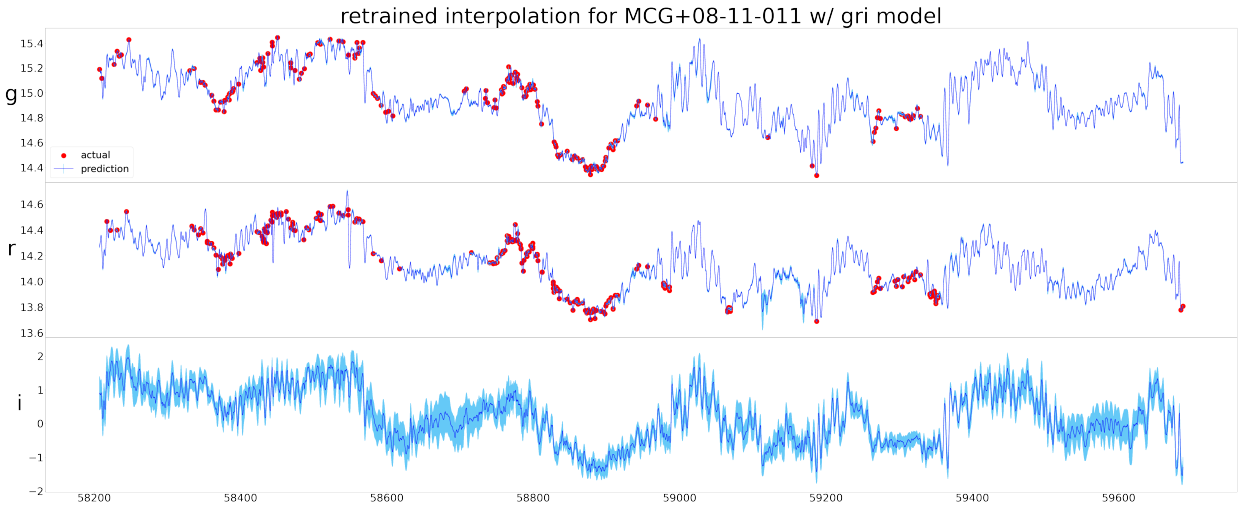
As far as fine-tuning goes, having run approximately 50k additional iterations with the trained *gri* model on the light curves for this object plus its associated resamples took the errors from around the average loss all the way down to a NLL of ~ -2.2 and MSE of ~ 0.002. Quite unreasonably, we can see what the model does with the missing light curve for i in 4.20a. Here the model exclusively uses its intuition from the g and r band light curves and its understanding of the relationships its learned between bands from other objects across the dataset to make its informed guess as to what the i band looks like. Otherwise, the model's interpolations fit the multivariate example well in that they are able to match the variability on short time scales (see figure 4.20b), while keeping a decent overarching shape across all of the light curves and assuming a stochastic

variability behavior. Yet the error snake is blatantly not very sensible because in instances where there are not points, across any of the light curve bands for that matter, the uncertainty does not increase (but at least the model knows to project more uncertainty in i's interpolation). We note that this object was filtered out of the original training set because it did not satisfy the minium length requirement of having 25 observations in i, and thus its being fine-tuned on showcases the models ability to adapt to unseen data.

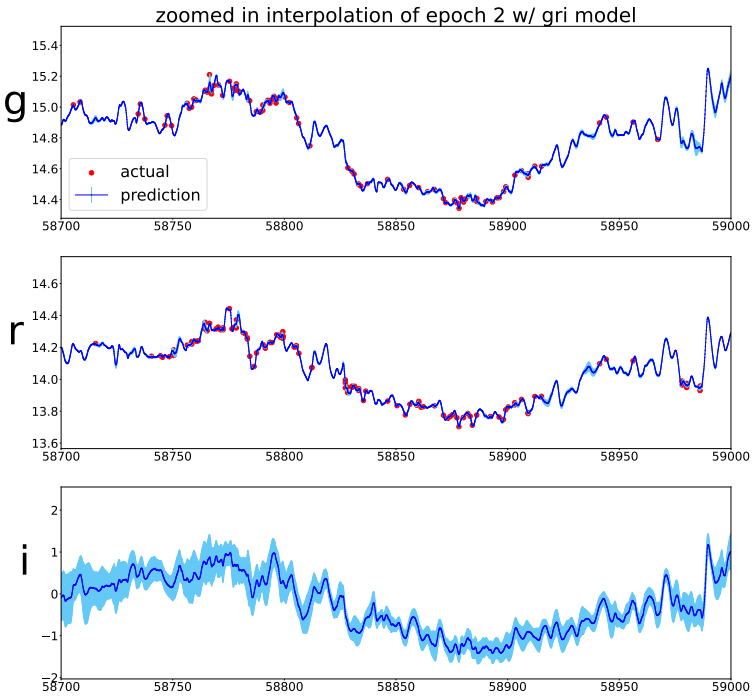
Before we move to calculate the lags, we also show the interpolations resulting from fine-tuning the g and r univariate models on MCG+08-11-011's g and r light curves respectively in figure 4.21a to show how they differ from the multivariate model in figure 4.20. In this case, the model is not able to take information from the light curves of the other filters to support its interpolations. Additionally, we show how the learned kernel for a multivariate model trained from scratch (exclusively on this example/its resamples) returns different results than those from the fine-tuned models in figure 4.21b.

We now pass the segments (1 and 2) and their associated interpolations from HeTVAE into PyROA, dropping the cadence of the interpolation from 0.1 days to 1.0 to save compute time. Reiterating, the median cadence of the g band light curve before the interpolation was about 3 days and the r, about 2 days. Both segments were about 300 days long individually and thus now have about 300 points each from the interpolation.

Pre-interpolation, the lags estimated by PyROA on epoch 1 were confidently in agreement across all of the different blurring window priors to estimate the r band lag at ~ 5 days, specifically for the default window it was $5.160^{+0.625}_{-0.621}$ days. We show the fit with the default priors in figure 4.22a. Post-interpolation, the lags returned were $0.161^{+0.071}_{-0.078}$, $1.924^{+0.581}_{-0.658}$ and $3.752^{+1.083}_{-0.894}$ days for the default (figure 4.22b), medium (figure 4.22c), and high (figure 4.22d) blurring priors respectively. The default prior overfit the light curves, tending the corresponding lag estimate toward zero. The medium prior visually appeared to have the best fit to the light curves, while the high prior underfit them. Nonetheless all of these estimates are more constraining to the reference lag of $0.69^{+0.16}_{-0.15}$ days than the

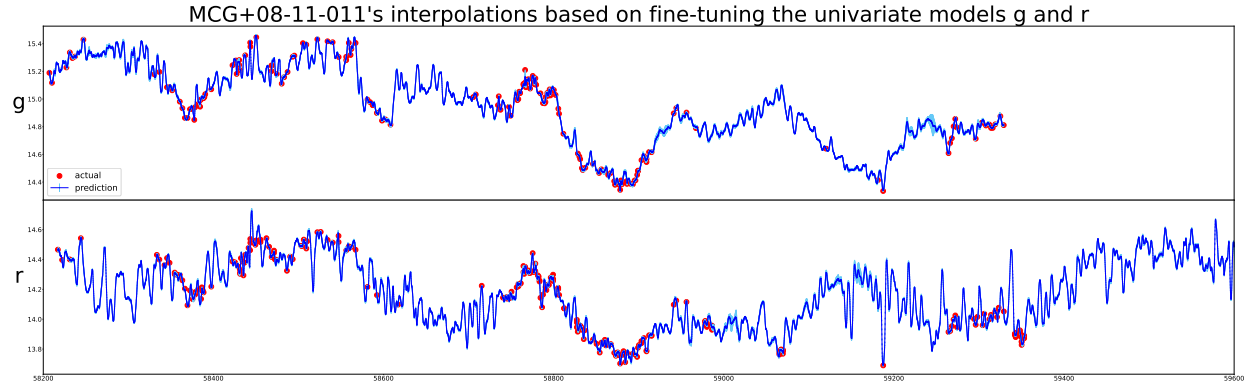


(a)

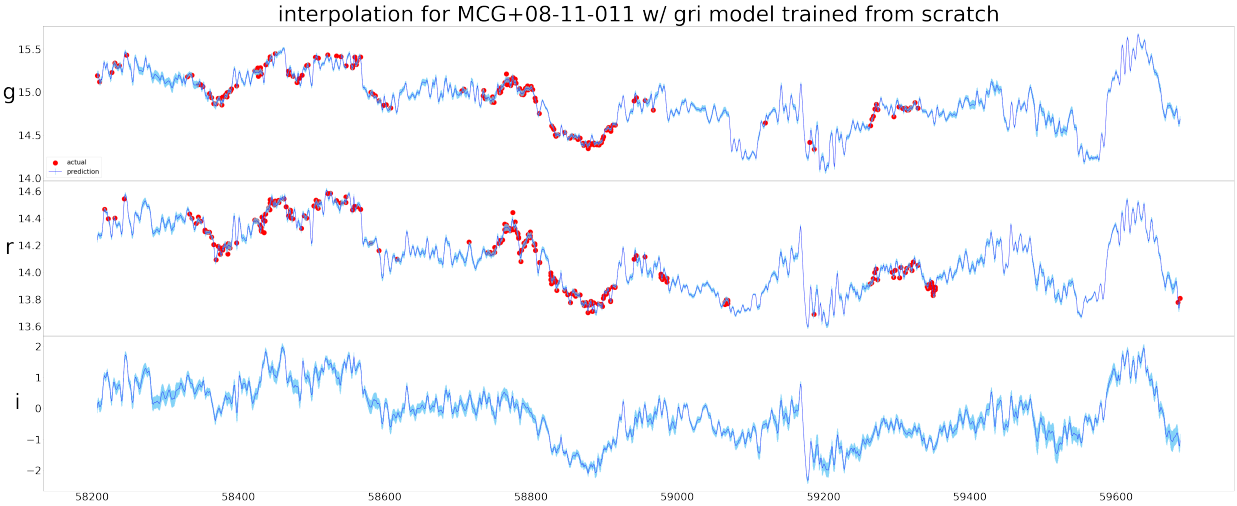


(b)

Figure 4.20: HeTVAE's fit to MCG+08-11-011's light curves after fine-tuning (0.1 day cadence), including a zoomed in view of the interpolation for epoch 2 in (b).

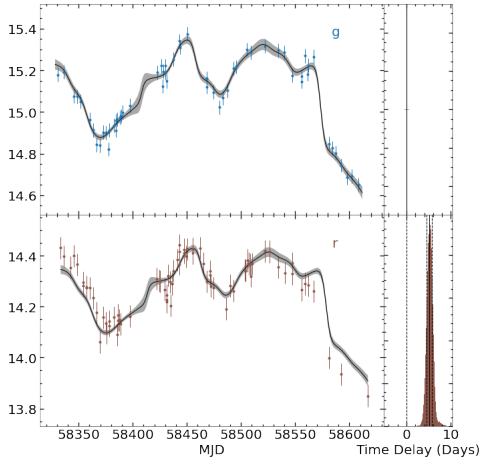


(a)

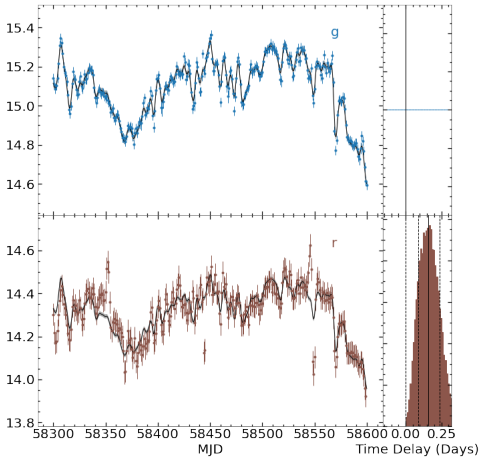


(b)

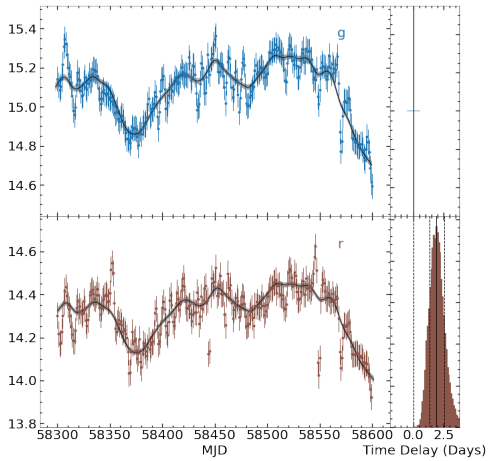
Figure 4.21: (a) shows interpolating the g and r band light curves with their associated univariate g and r models. It is quite clear between 58200 and 58400 MJD that the interpolations are contrasting because the g model does not have knowledge of r light curve and vice versa, whereas the model knows they tend to be aligned with one another in the fine-tuning with the multivariate model (figure 4.20). (b) shows training a *gri* model from scratch with just the light curves of MCG+08-11-011 (plus their resamples).



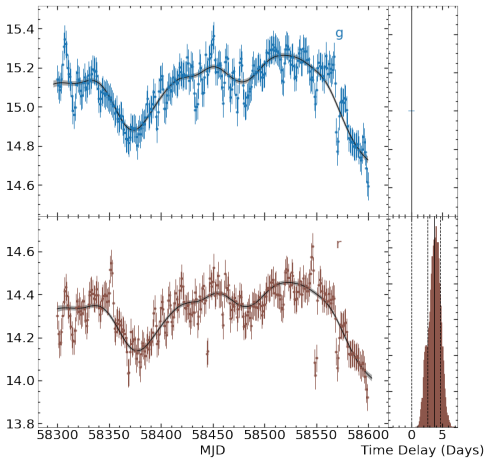
(a) Pre-interpolation with default blurring.



(b) Post-interpolation with default blurring.



(c) Post-interpolation with medium blurring.

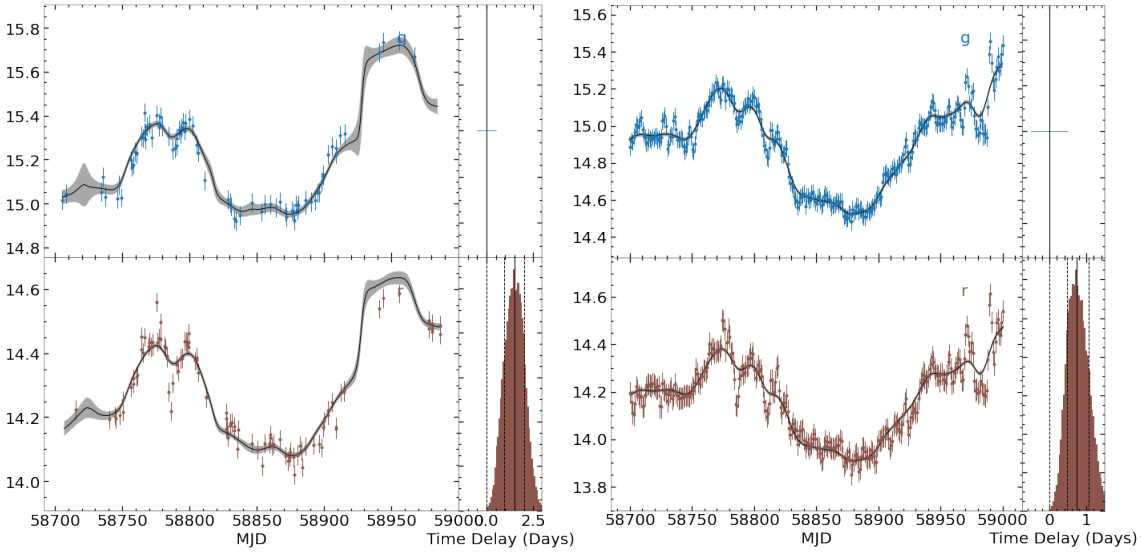


(d) Post-interpolation with high blurring.

Figure 4.22: MCG+08-11-011's PyROA fits on epoch 1.

pre-interpolation estimates.

As for the raw segment of epoch 2 the lags were all generally in agreement (all between 1.5 and 2 days) across the different blurring window priors. The fit with the default priors is shown in 4.23a as it visually was an adequate fit and had the longest and skinniest of the posteriors of the three. The r lag in that case was $1.528^{+0.428}_{-0.399}$ days. Post-interpolation, the default blurring prior again overfit the curves and tended the lag towards zero ($0.040^{+0.029}_{-0.045}$ days), while the medium blurring seemed to be the best fit, returning $0.744^{+0.273}_{-0.319}$ days. Just as in epoch 1, the highest blurring window underfit the interpolated light curves and returned a lag of $1.636^{+0.551}_{-0.573}$ days. For this segment, the interpolation resulted in



(a) (b)

Figure 4.23: Pre (a) and (b) post-interpolation PyROA fits for epoch 2 of MCG+08-11-011's ZTF light curves.

the estimated lag being within the uncertainty limits of the reference lag, i.e. $0.744^{+0.273}_{-0.319}$ versus $0.69^{+0.16}_{-0.15}$ days.

Finally, we are whimsically curious to try to extract lags from the i band light curve that HeTVAE interpolated without having had any points, but just the knowledge of the other two dimensions (g and r) and the knowledge from other object's/example's interdimensional relationships using the interpolation associated with the second segment. To reduce bias, we set the transfer function to be Gaussian, not log-Gaussian, so that negative lags are possible. Additionally, we have to lower the prior range for the mean of the i band light curve because it is not unnormalized. i's initial τ is set to be 0 and its prior range -50 to 50 days. We show the fit with medium blurring in 4.24. The estimated lags in this instance were $0.951^{+0.289}_{-0.284}$ and $0.484^{+0.309}_{-0.392}$.

We show the reference lags and lag estimates from the best PyROA fits pre and post interpolation of the epochs in 4.1.

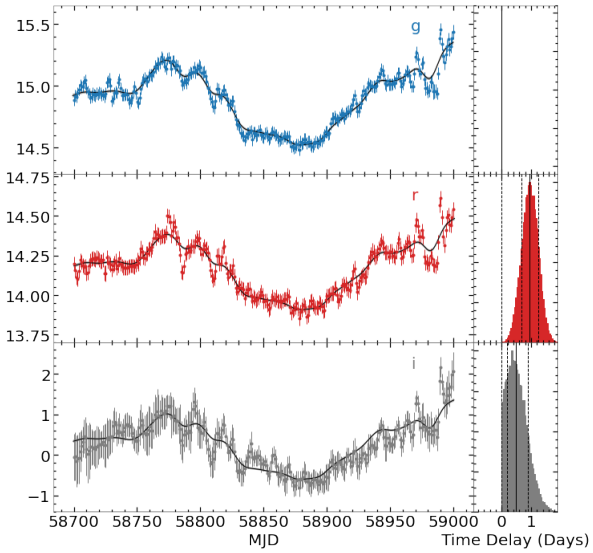


Figure 4.24

		MCG+08-11-011		
		τ_{gg}	τ_{gr}	τ_{gi}
Reference (javelin)		$0.00^{+0.08}_{-0.07}$	$0.69^{+0.16}_{-0.15}$	$1.02^{+0.20}_{-0.18}$
PyROA				
epoch 1	original	0.	$5.160^{+0.625}_{-0.621}$	−−
	interpolation	0.	$1.924^{+0.581}_{-0.658}$	−−
epoch 2	original	0.	$1.528^{+0.428}_{-0.399}$	−−
	interpolation	0.	$0.744^{+0.273}_{-0.319}$	−−
	interpolation (including i)	0.	$0.951^{+0.289}_{-0.284}$	$0.484^{+0.309}_{-0.392}$

Table 4.1: Lags garnered from PyROA with ZTF light curve segments from MCG+08-11-011.

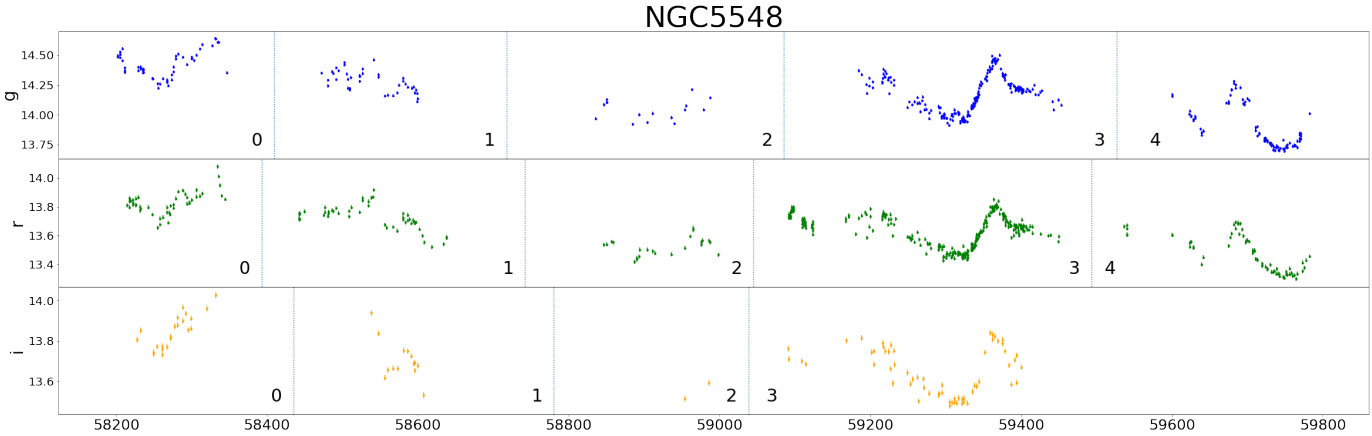


Figure 4.25: Epoch separations for NGC 5548's ZTF light curves.

4.4.2 NGC 5548

The light curves for NGC 5548 with their epoch separations are visible in figure 4.25. We focus in on epoch 3 (59150 - 59450 MJD) in this analysis. We can observe that the i band in this instance has enough observations to match the variability features of the other two light curves in epoch 3. We again interpolate them to a uniform 1 day cadence, but note the resolution of the original ZTF light curves for the g and r bands is quite high already with median cadences of 0.996 and 0.988 days, whilst i 3.971.

The reference lags, as published in [74], are from SDSS ugriz light curves with an approximately daily cadence over seven months ($\sim$ 200 days) from January to July 2014. As such, we have comparable cadences in the g and r bands whereas the i band is, not unusually, lacking for the ZTF data, but on the other hand we have a larger window of observations. We again have to subtract out the estimated g lag from the reference lags as it is our driving light curve. The reference lags were also computed with Javelin, which we know linearly detrends light curves by default, and thus we do the same for consistency's sake. In this occurrence, the published lag estimates are not unreasonably smaller but rather on par with the cadence of the light curves, being $0.93^{+0.06}_{-0.07}$ and $2.01^{+0.11}_{-0.08}$ for the r and i bands respectively.

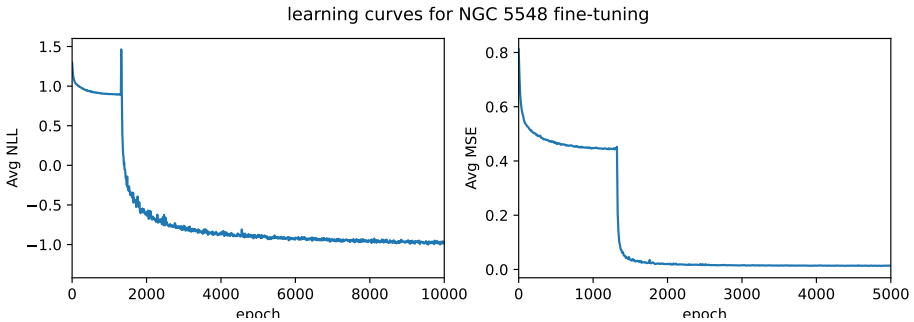


Figure 4.26: The average MSE and NLL values while fine tuning the *gri* model on the ten resamples of NGC 5548's light curves. The first part of the curve, from 0 to about 1500 iterations shows the the model's training on the full dataset before switching to exclusively the resampled light curves of NGC 5548. Most of the relearning happens within 500 iterations, but the NLL continues to converge until around 50k iterations, and the MSE converges within the shown 5k iterations.

The retraining process converged at about 50k iterations and brought the NLL to -1.245 and the MSE to 0.010, but the mass of the improvement happens in about 500 iterations of fine-tuning. We show the learning curves for the first several thousand iterations of the retraining process in 4.26 to support this. The interpolations for epoch 3 from fine-tuning are shown in figure 4.27, which shows how it fits the short timescale variability patterns of the light curves.

In both the interpolations and original versions of the light curve portions, we obtain 50k samples of the PyROA's MCMC chain and take the last 5k as the posterior to ensure as best convergence as we could muster. For epoch 3 of the original light curves, we found very similar estimated lags for both the default and medium blurring priors, but a much shorter and stubbier posterior for i in the medium blurring window. Any higher blurring resulted the curves being underfit. We thus show the corner plots for the default window prior (figure 4.28a). Here, the mean lag and associated uncertainty returned for the r band was $2.209^{+0.628}_{-0.327}$ and $1.634^{+0.616}_{-0.591}$ for i. But noteworthily, there is a second small peak in the posterior, also apparent in the medium blurring configuration, that skewed the mean lag of r from the centroid estimate of 2.145 days; it is seemingly aligned with 1.211 days, the 16th percentile of the distribution (the second dashed line from the left of 4.28a).

Upon using the interpolation for this region, which also returned very similar lag estimates

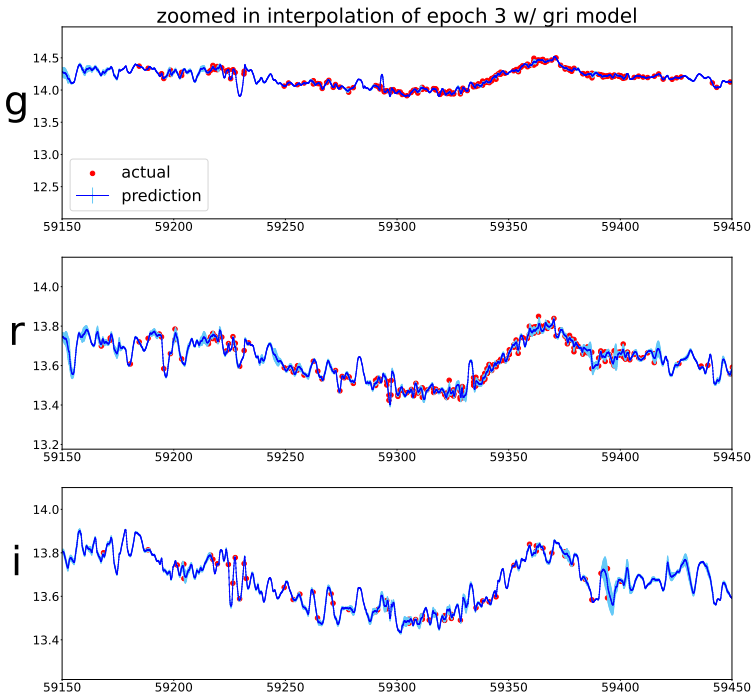


Figure 4.27: View of the full 0.1 day cadence interpolation of epoch 3 of NGC 5548's ZTF light curves resultant from fine-tuning.

for either using the default or medium blurring and was significantly underfit with the higher blurring prior, the r lag posterior centers instead near the value that was at this second peak, $1.317^{+0.370}_{-0.402}$, shown in figure 4.28b, while the i lag returned $1.606^{+0.374}_{-0.314}$, a similar value as before using the interpolation but with lower uncertainties. The final time-delay estimates from the original light curve portions and associated interpolations are shown in table 4.2.

NGC5548				
		τ_{gg}	τ_{gr}	τ_{gi}
Reference (javelin)		$0.0^{+0.06}_{-0.04}$	$0.93^{+0.06}_{-0.07}$	$2.01^{+0.11}_{-0.08}$
PyROA				
epoch 3	original	0.	$2.209^{+0.628}_{-0.327}$	$1.634^{+0.616}_{-0.591}$
	interpolation	0.	$1.317^{+0.370}_{-0.402}$	$1.606^{+0.374}_{-0.314}$

Table 4.2: Final lag estimates for epoch 3 of NGC5548.

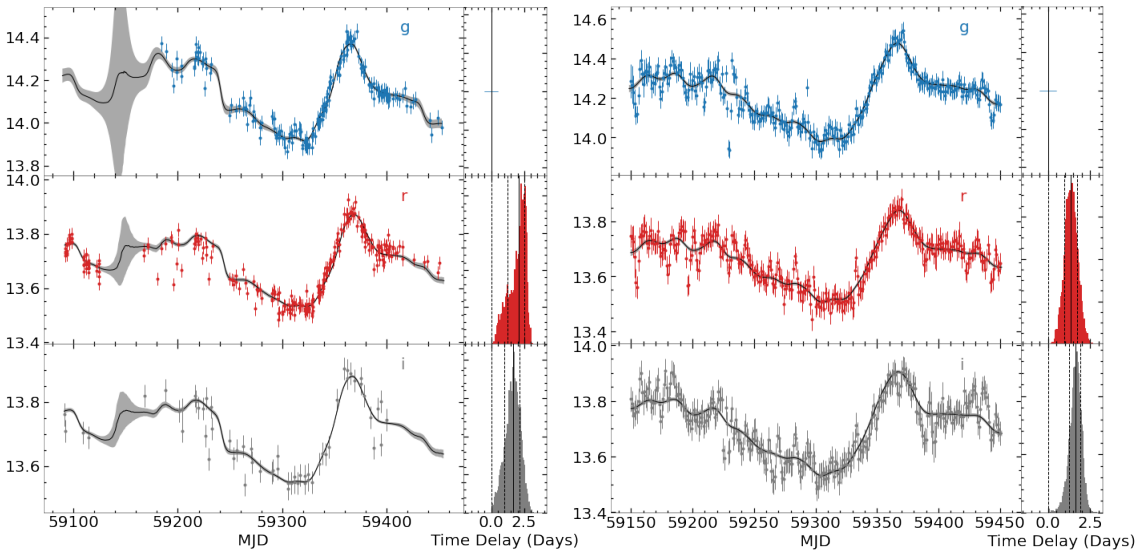


Figure 4.28: PyROA fit and posteriors for the estimated time-delays (a) pre-interpolation (default blurring priors) and (b) post-interpolation (medium blurring).

4.4.3 3C 273

The light curves for 3C 273, with their epoch separations shown in 4.29, have an average cadence of 1.965, 1.848, and 5.471 for the g,r and i bands respectively. We use the only pronounced i band segment labeled as epoch 1 and epoch 3 of the g and r bands in this anaylsis. The reference lags we use are from James Thorne's (Durham) master's thesis [73] and are soon to be published, where Las Cumbres Observatory (LCO) data with an average cadence across the *bgvriz* filters as 0.583 days (and less than five minutes in 75 percent of the data) and observations across 58500 to 59700 MJD were employed to estimate lags. Our ZTF data actually overlaps entirely with this, being from 58200 to 59900 MJD. We also know the reference lags were calculated with PyROA, that their concluded lags were garnered from 59150-59450 MJD, which fortuitously aligns with our third epoch, and finally that their data was detrended by subtracting a fitted linear polynomial prior to being passed into PyROA.

As such, we have our most controlled experiment being that the reference lags were estimated with PyROA and that we can use the same observational time frame with our ZTF data that was used to estimate the reference lags. Thus, this experiment is the

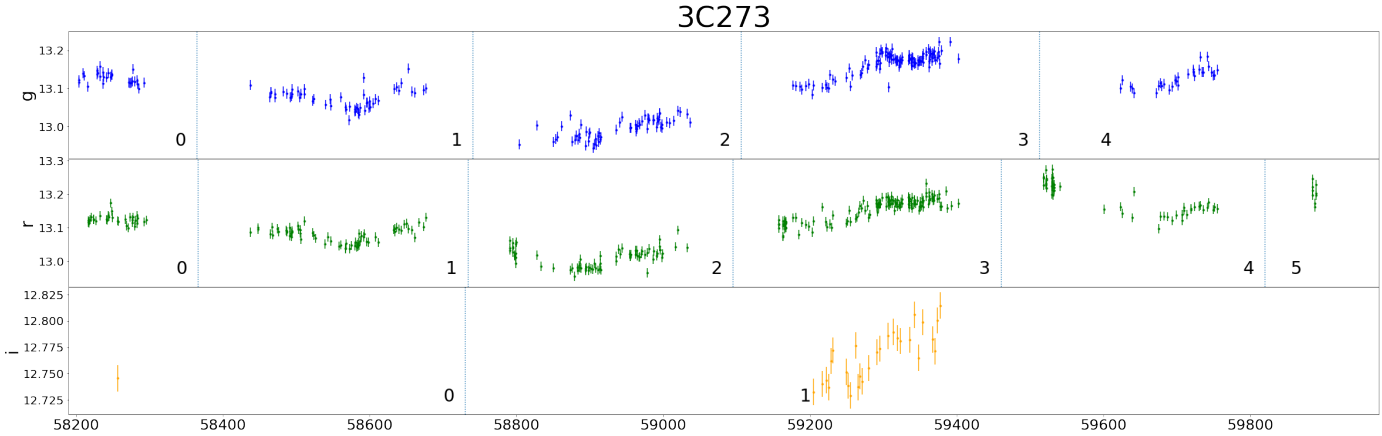


Figure 4.29: Epoch separations of 3C 273's ZTF light curves.

most direct comparison of how the lower cadence in ZTF data, especially low in i with 29 observations) affects the reverberation mapping analysis and how the interpolation might improve it. It is also of note that the variability features in the associated light curves of 3C 273 are the least pronounced as compared to the other objects we have analyzed, posing another issue for the reverberation mapping, but the reference lags are at least larger than the cadence ($11.01^{+1.71}_{-2.07}$ days for r and $13.94^{+2.94}_{-2.37}$ days for i).

The retraining process brought the loss down to an NLL of -1.373 and MSE of 0.0165, which again seemed to converge around 50k more iterations (but again the majority of improvement within 1k). We show the interpolations in 4.30. We focus in on the third epoch and detrend both the original light curve segment and the section of the interpolation associated with it (59150 - 59450 MJD; 1 day cadence) separately. Pre-interpolation, PyROA was unable to match the ROA shape to the i band light curve, returning very large uncertainties on its estimated lag and very non-uniform posteriors for both r and i, shown in 4.31a. Specifically, the r's lag posterior was right-skewed near 0, likely because lag estimates less than zero are cut off by the log-Gaussian transfer function, and the i's lag posterior was almost uniform. The assosciated estimates were $1.119^{+0.820}_{-1.635}$ and $26.068^{+14.836}_{-15.975}$ days for the r and i bands respectively. We also tried to fit the g and i band light curve portions without the r and noticed PyROA still was not able to fit the i band, returning an almost identical posterior.

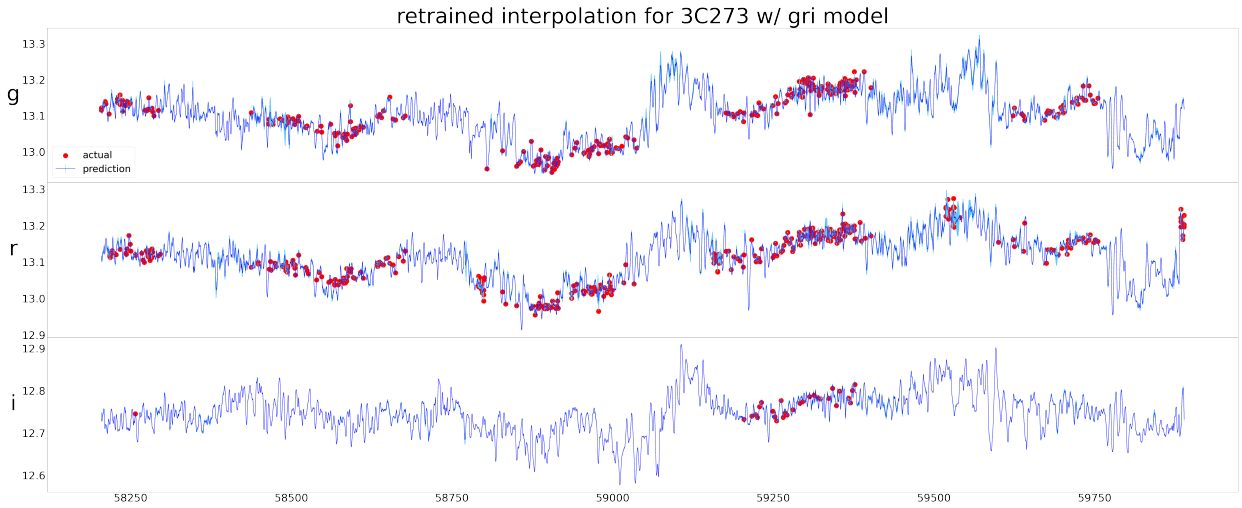
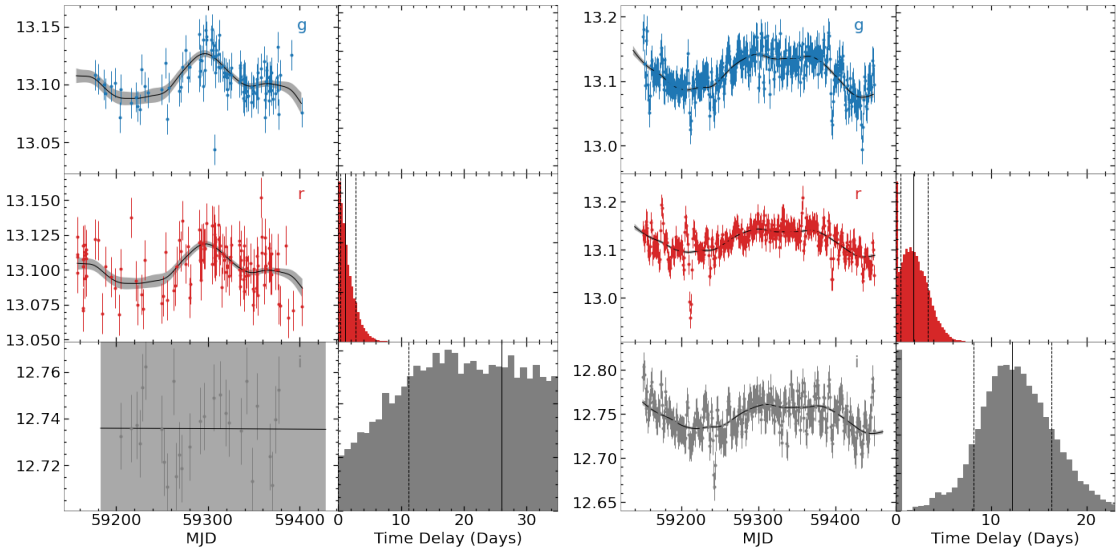


Figure 4.30

NGC5548				
		τ_{gg}	τ_{gr}	τ_{gi}
Reference (PyROA)		$0.0^{+1.77}_{-1.96}$	$11.01^{+1.71}_{-2.07}$	$13.94^{+2.94}_{-2.37}$
PyROA				
epoch 3	original	0.	$1.119^{+0.820}_{-1.635}$	$26.068^{+14.836}_{-15.975}$
	interpolation	0.	$1.843^{+1.322}_{-1.590}$	$12.206^{+4.039}_{-4.127}$

Table 4.3: Lags garnered from PyROA with ZTF light curve segments from 3C273.

Upon using the interpolation, the i band's fit was clarified, returning a clearer posterior whose lag estimate of $(12.206^{+4.039}_{-4.127})$ is very much aligned with the reference lag of $13.94^{+2.94}_{-2.37}$. As for the r lag, the interpolation raised the estimate slightly and returned a much more uniform posterior as well. The interpolation corner plots from the PyROA fit are shown in figure 4.31b, while the all of final lag estimates are shown in figure 4.3.



(a) (b)

Figure 4.31: Pre (a) and post (b) interpolation PyROA fits for epoch 3 of 3C 273's ZTF light curves.

4.4.4 Discussion

Upon inspecting the light curve interpolations, we found that long-term variations were matched, but more rapid variability was not covered, however the fine-tuning procedure corrected this. The multivariate model was able to make educated guesses on gaps that were shared across dimensions, while understanding differences between dimensions like variability amplitude increasing with lessening wavelength (i.e. quite explicit in 4.27). Moreover, the network picked up on the stochastic variability patterns without explicit direction, yet the uncertainties returned were not very intuitive. The adaptability of the model was shown by way of this fine-tuning procedure, and in this vein we mention that we were able to also re-train explicitly on the shorter segments of the light curves (and not just cut out these segments from the fine-tuned interpolations of the whole light curves), which could have been an issue because of the way the network uses reference points, and those are likely very settled to the range of the whole light curves.

For all three of the objects covered, we were able to hone in the estimated time-delays more closely to the published estimates from higher-cadence surveys. MCG+08-11-011 was an example where we had access to two 300 day observational epochs that highlighted

very pronounced variability features, at least for the g and r bands, but of course had the relatively low-cadence characteristic of the ZTF survey. In the case of epoch 1, r's time-delay was able to be constrained from $5.160^{+0.625}_{-0.621}$ to $1.924^{+0.581}_{-0.658}$ days using the interpolation from HeTVAE, where the reference lag was $0.69^{+0.16}_{-0.15}$. For epoch 2, the interpolation improved the time-delay estimate of r from $1.528^{+0.428}_{-0.399}$ to $0.744^{+0.273}_{-0.319}$, making it align with the reference lag. Finally, we used the interpolation from HeTVAE for the i band, which had no actual observations, and changed the prior settings for PyROA to allow for a very broad range of lags (-50 to 50 days) and found the lag was able to be estimated from the interpolation and was non-negative, which reflects how the power of a multivariate model although the lag was not longer than the r band lag so we can not presume that the network has learned that much about the intra-band relationships.

As for the reverberation mapping analysis of NGC5548, we looked closely at epoch 3, where we had good variability structures across the three bands, but of course an imperfect cadence especially lacking for the i band. We found that pre-interpolation PyROA returned a multimodal posterior for the r band lag estimate, where the primary centroid did not align with the reference lag, but the secondary centroid did, and post-interpolation, the posterior for r's lag instead was explicitly centered where the secondary centroid had been. The lag estimate for i based on the interpolation did not change its value at all really, but rather confined the uncertainties.

3C 273 was an example where we had a very noisy i band portion of the light curve to use and less variability structures as compared to the other objects, in addition to having the standard low-cadence of observations. Here, PyROA was very thrown off for both lag estimates, returning obscurely shaped posteriors and a non-existent fit for the i band. The interpolation made it so that both of these posteriors were cleaned up and the i band lag estimate was in good agreement with the reference lag.

In all of these instances, the eldest data that was used to estimate the reference time-delays was from 2014. Accordingly, it is unlikely that the accretion disc of the three

objects has changed since then, meaning the true lags would also likely not change. In NGC5548 and MCG+08-11-011, Javelin was used to estimate the lags, which employs a DRW model to fit the curves (and which we should have probably also used in these experiments as a comparison to see what lags could be estimated pre-interpolation of the ZTF light curves). In literature [76], [28], [77], it has been discussed that Javelin lag uncertainties are somewhat tight. If the lag uncertainties for the reference lags that were computed using Javelin (NGC5548 and MCG+08-11-011) were higher, it could make some of our lag estimates from the ZTF data more agreeable.

Moreover, we note that the error bars from the interpolation were likely underestimated, and were helped by PyROA's rescaling algorithm. If we look at the rescaling/inflating of the errors done by PyROA on its fits to, for example, MCG+08-11-011 pre and post interpolation, for segment 1 we get 0.0312 pre-interpolation and 0.06362 post for the g band and 0.0422 pre-interpolation and 0.0599 post for the r band, while for segment 2, 0.0417 pre-interpolation and 0.0444 post for the g band and 0.0277 pre-interpolation and 0.0453 post for the r band. Similar discrepancies for the error inflation parameter between pre and post interpolation fits from PyROA were seen with both NGC 5548 and 3C 273 so it is safe to assume that the error bars from the interpolation of HeTVAE were underestimated a bit more than those of the original light curves; a flaw we could visually point out of the interpolations, luckily corrected by PyROA. We presume that training the the augmented loss function to include the observational error would improve the uncertainty estimates output by the interpolations, but also we probably should have stopped the fine-tuning earlier so that the interpolations were not as entirely overfit as they were (although that is somewhat the point of fine-tuning).

We additionally noticed that when fitting PyROA on the interpolations, the algorithm most always settled its blurring window parameter Δ to the bottom end of the prior setting, meaning that it was eager to fit the interpolations as tightly as possible. Thus, we were very in control of how closely the fit was via these priors. We can assume it acted in this way because the ROA was significantly penalized by the χ^2 term such that

this penalty outweighed the penalty for having an overly-complex model because there were so many points to fit from the interpolation. The danger here is that when PyROA overfit the interpolations, the lags tended to zero because the light curves became too indistinguishable (see figure 4.22b in MCG+08-11-011's analysis).

In the immediate future, we hope to try higher cadences of the interpolations (0.5 days or less) instead of the 1 day cadence that was used. We also hope to try estimating lags by interpolating larger gaps (like those between epochs). It would also be relevant to have time-delay estimates from Javelin on the pre-interpolated light curves to see how HeTVAE's interpolation compares to the DRW fit in terms of their affect on lag estimates. Moreover, it would be apt to study the power spectrum of these light curves to better understand the prominence of different variability timescales acting based on the interpolations and compare it to the kernel of a damped random walk or damped harmonic oscillator. And of course it would also be beneficial to use a larger dataset so that we can be more certain the larger distribution of AGN is covered.

Finally, it is relevant to discuss the scalability of this procedure as it would be desirable to perform the same such lag estimations on significantly more than three objects. What restricts this and slows things down is both the fine-tuning procedure and using PyROA. The quick-and-dirty method would instead be to use the interpolations as is (i.e. like those in 4.2) and employ ZDCF/ICCF instead with the hope that the time-delays resultant of this procedure would still help constrain lags better than without the interpolation; even manually picking out segments/epochs of the light curves is a hinderance to efficiency. Omitting the use of PyROA, but still fine-tuning would be an improvement because fine-tuning is not too much of a hinderance due to the efficiency of gpus. We leave this testing for future work, but deemed it worth acknowledging.

CHAPTER 5

Conclusion

The original project outline for this master's was to use and analyze existing algorithms for AGN light curve fitting and reverberation mapping. We instead had the naïve idea to find our own method and were ultimately drawn to the deep learning regime, having a handful exemplar papers ([55], [38], [39], [40], [56], [43], [42]) of neural networks being applied to astronomical time series to guide our methodologies. The interesting case of time series data is that it is adjacent to the sequential modeling used for language, which there are hoards of information and blog posts about, and the research is further along in those contexts as per their commercial use cases and profitability. Because of this, most of our understanding of the new bells and whistles in deep learning comes from that angle and guides chapter 2. In it, we discuss and try to close the gap in our understanding between two major paradigm shifts from recurrent models (LSTM, [31]; GRU, [32]) to Transformers [3], from standard autoencoders to the variational variety (VAEs; [36]) and from unsupervised learning to self-supervised learning.

Once we better understood some of these newer paradigms, we shopped around for a model that could be applied to irregular time series and found that this is a not so trivial hiccup to account for considering there were so few to choose between. We landed on

'Heteroscedastic Temporal Variational autoencoder' (HeTVAE; [1]). Its attraction and novelty on paper, as compared to other applied deep learning models on light curves, was that it applied all of these up to date concepts, not just one or two, as well as considered uncertainty, could provide intepolations and could handle the light curves in their multivariate form. For instance, in comparing it with the GRU-based VAE (a GRU) in [40] that was used for anomaly detection of AGN light curves, HeTVAE is attention-only rather than recurrent, it employs self-supervised learning as opposed to unsupervised learning in that it does not encode the whole light curve and try to predict the same light curve after encoding/decoding, but instead it encodes a subset of the light curve's points and tries to predict to those not encoded. In addition, there was no KL annealing schedule employed in this GRU-VAE, not to mention it only considered univariate light curves and is not capable of probabilistic interpolation.

In practice, HeTVAE required some nurturing and frankly we have not had significant hands-on experience with training/working with these models so a lot of the code we wrote/rewrote to adapt it was in an attempt to help our self-understanding of its inner-workings, i.e. we wrote a significant amount of code to use the model (interpolate with-/decode/encode with/plot results) in a way that made more sense to us. There were some bugs of course likely because the authors were more focused on the publication and how the code needed to run for their datasets/their configurations like bugs in the multi-head attention (1 head sufficed for their datasets), dropout, etc. A larger adaptation of ours was for the astro datasets, which of course are not downloadable and formatted benchmark datasets. We were truly making our best educated guesses as far as writing code to format and prepare the light curves in a multivariate form (described in appendix A) because we had very little examples to base the procedures off of, i.e. How do you normalize multivariate time series? How do you format the multivariate time values within the numpy arrays in a neural network-digestible form? We also made our best guess on augmenting the loss function to include information of observational error (even though we left training with it for future work). Other things we added to supplement HeTVAE, beyond the authors, included employing the KL annealing schedule, using uniformly

spaced arrays instead of the union of all time points across the dataset for n_union_tp (this was actually left for future work by HeTVAE's authors), readying the code for the HPC system and the ability to save/load model checkpoints. We mainly hope that the work we have done created a better interface to train/access/use the model and urge the reader to gander at the github repository (`https://github.com/mwl10/hetast`).

Now training the model was interesting because of the amount of switches/controls we had access to; there were an innumerable amount of different configurations for this model's hyperparameters compounded by the amount of different configurations that modern deep learning packages offer, i.e. different learning-rate schedules, optimizers, etc. We found ourselves stubbornly spending too much time tinkering with the software as opposed to more importantly doing the science. When we made changes we often would introduce bugs that slipped past us, and thus we came to a deep understanding of the importance of writing tests in the future. We deem it a weakness of HeTVAE to have so many configurable hyperparameters, but nonetheless once the model was configured it was able to learn on a plethora of multivariate light curve examples, including entirely missing light curves for a given band in an example, which is an important case as filtering out these examples for larger datasets means losing out on a lot of data. We also found we were able to retrain models at will, which turned out to be very convenient for the interpolation's sake as the model did not converge at very low losses.

In using the model, we were able to show off the intuitive kernel learned by the latent space in a few ways. We visualized the attention scores to see what the first reference point was looking at when gathering what information from the light curve should be passed to it (figure 4.3) and provided simulated light curves with a couple of different model iterations. We confirmed the space's continuous nature via the blending experiments (figure 4.4) and the morphing across bins of the reconstructions made from the PCA binning experiments, i.e. figure 4.8. More study of the reconstructions of bins from higher principal components would be interesting, i.e. what other discrepancies across light curves have been learned by the latent space? As well as more study of the attention

scores, like what timescales are being paid most attention to?

What we learned from the PCA experiments was the danger of algorithmic bias (i.e. a model performing better/worse on a given subset/group within a dataset) in that there was a disproportionate number of less-than-variable AGN examples in the dataset, which pushed all light curves opposite of that description to be anomalous. Thus we created an algorithm that was able to filter out the more variable light curves, but were not able to target anomalies within the variable light curves themselves. We could use the algorithm in the future to curate a dataset of more variable light curves and then train a new model explicitly on those light curves. Then we would want to counter other biases from parameters we can calculate (i.e. physical parameters of the black hole, light curve descriptors, etc) so they would not distract from 'truthful' anomalies. The irony is that filtering out examples means losing information from the wider AGN distribution, which is why countering bias within the objective function is helpful. Regardless, algorithmic bias is a topic that is coming more to the forefront of AI research, and we are eager to keep an ear on the ground.

In the reverberation mapping experiments, we successfully improved time-delay estimates from the low cadence ZTF light curves by utilizing interpolations resultant from fine-tuning a multivariate model on three specific objects, 3C 273, MCG+08-11-011 and NGC 5548, and resamples of their light curves to add observational error information. We used PyROA as a way to show what a modern reverberation mapping algorithm was able to estimate without our interpolations (using solely its running optimal average) and subsequently with our interpolations, as compared to published reference lags for these objects as calculated from higher resolution light curves. We found that in none of the instances the interpolations worsened the lag estimates, and in most instances made them more aligned with the published estimates. We additionally showed the extent to which the model learned the transfer function given its interpolation of the i band light curve of MCG+08-11-011 without having had any actual observations to make use of and found that we were able to calculate a non-negative lag with the interpolation. We note that

to further gauge the novelty of using HeTVAE's interpolations to improve time-delay estimates we should compare results to those of Javelin (a DRW fit) on the raw ZTF light curves. It would be interesting also to use the newer damped harmonic oscillator (DHO) fits, which are pretty easy to get with *EzTao* software [78], to estimate lags with and compare them with our interpolations.

In the future, we hope to perform experiments similar to [39], i.e. find the features of the latent space most associated with different black hole parameters and change/exaggerate their values to see how it changes the light curve reconstructions, i.e. how does an increment/decrement to black hole mass change the light curve? This is especially regrettable because doing this with the multivariate model means we could have studied how the light curves change across the different filters given different black hole parameters, i.e. how does the transfer function change for different sorts of black holes?

The weaknesses of HeTVAE are in its vast number of hyperparameters and in its uncertainty estimates for the interpolations. Asking the model to predict uncertainty in addition to a point estimate is likely asking a lot of the network, at least for these stochastic high-dimensional light curves. There is no reason HeTVAE's architecture be set in stone given that model fails to include residual connections as well as batch normalization, which are proven to help with training Transformers and might improve its performance. Either way, it is in this vein that we think the future of Neural Processes [50] is promising as these models frame the problem of: learning a function over functions in a way that does not require all of the 'tricks' that HeTVAE employs, on top of the fact that they more cleanly handle uncertainty, i.e. as compared to including observational uncertainty in HeTVAE, where augmenting HeTVAE's loss function was cumbersome (equation 3.15/3.16). Specifically, models in the neural process family we have not seen tested are the convolutional variety. For instance, the Convolutional Latent Neural Process [53] that introduces a powerful inductive bias: translational equivalence to neural processes. In addition very powerful generative models like normalizing flows and diffusion/denoising models (probably the most powerful) have been introduced as

evolutions of VAEs. For example, while VAEs inexplicitly optimize the log-likelihood of the data by maximizing the evidence lower bound (ELBO) and have a strong Gaussian prior, the later two explicitly learn the data distribution and therefore the loss function is simply the negative log-likelihood, allowing for more flexible/multimodal distributions. Of course, these models require much more compute and likely do not have adaptions for irregular time series yet, but hopefully approaches to handle missing data in general are more standardized moving forward. They are most noteworthily used in image generation (i.e. caption-to-image generators), but as is usually the case the scientific adaptations are made after the marketable ones. We believe that in the future there will be deep learning models that adapt the intuitive framing of these time series datasets as stochastic processes (like neural processes do) to the advanced generative capabilities of diffusion models and normalizing flows.

Using more advanced dimensionality reduction techniques (for visualization) than PCA would probably also be fruitful to more adequately describe what is happening in the latent distribution. PCA lacks in its ability to preserve local clusters, but using newer visualization methods that aim to do this nonlinearly like Uniform Manifold Approximation and Project (UMAP; [79]) would be interesting to use in order to better our interpretation of the latent space and thus AGN subgroups. Unfortunately these methods require tuning more hyperparameters; in UMAP's case: nearest neighbors, which is the number of expected neighbors within a cluster and minimum distance, which describes how tighly UMAP packs close-together points.

All in all, this work has configured Heteroscedastic Temporal Variational Autoencoder (HetVAE) for an irregular, multivariate time series dataset consisting of AGN light curves from ZTF across g, r and i bands for several thousand objects. HeTVAE's ability to account for inhomogenous datasets and its relative training robustness given its complexity was shown, being the first method to provide encouraging results for creating multivariate summaries of AGN light curves and nonlinearly interpolating each light curve respective to the summary, which will be especially relevant as we look to maximally harness the

6 filter light curves incoming from the Vera Rubin Observatory Legacy Survey of Space and Time (LSST).

Bibliography

[1] S. N. Shukla and B. M. Marlin, "Heteroscedastic temporal variational autoencoder for irregularly sampled time series," 2021. (document), 5

[2] E. M. Cackett, M. C. Bentz, and E. Kara, "Reverberation mapping of active galactic nuclei: From x-ray corona to dusty torus," *iScience*, vol. 24, p. 102557, jun 2021. (document), 1.2

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. (document), 2, 2.5, 2.1, 2.6, 2.2.1, 3.1.3.1, 5

[4] M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami, "Conditional neural processes," 2018. (document), 2.3, 2.7

[5] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin, "Cyclical annealing schedule: A simple approach to mitigating kl vanishing," 2019. (document), 3.7, 3.2.2.1

[6] I. Jankov, A. B. Kovač ević, D. Ilić, L. Č. Popović, V. Radović, I. Čvorović-Hajdinjak, R. Nikutta, and P. Sánchez-Sáez, "Photoreverberation mapping of quasars in the context of legacy survey of space and time observing strategies," *Astronomische Nachrichten*, vol. 343, nov 2021. (document), 3.10

[7] P. M. Rodríguez-Pascual, D. Alloin, J. Clavel, D. M. Crenshaw, K. Horne, G. A. Kriss, J. H. Krolik, M. A. Malkan, H. Netzer, P. T. O'Brien, B. M. Peterson, G. A. Reichert, W. Wamsteker, T. Alexander, P. Barr, R. D. Blandford, J. N. Bregman, T. E. Carone, S. Clements, T. J. Courvoisier, M. M. De Robertis, M. Dietrich, H. Dottori, R. A. Edelson, A. V. Filippenko, C. M. Gaskell, J. P. Huchra, J. B. Hutchings, W. Kollatschny, A. P. Koratkar, K. T. Korista, A. Laor, G. M. MacAlpine, P. G. Martin, D. Maoz, B. McCollum, S. L. Morris, G. C. Perola, R. W. Pogge, R. L. Ptak, M. C. Recondo-González, J. M. Rodríguez-Espinoza, E. L. Rokaki, M. Santos-Lleó, K. Sekiguchi, J. M. Shull, M. A. J. Snijders, L. S. Sparke, G. M. Stirpe, R. E.

Stoner, W. H. Sun, S. J. Wagner, I. Wanders, J. Wilkes, C. Winge, and W. Zheng, "Steps toward Determination of the Size and Structure of the Broad-Line Region in Active Galactic Nuclei. IX. Ultraviolet Observations of Fairall 9," *The Astrophysical Journal Supplement Series*, vol. 110, pp. 9–20, May 1997. 1.2

[8] S. Vaughan, R. Edelson, R. S. Warwick, and P. Uttley, "On characterizing the variability properties of X-ray light curves from active galaxies," *Monthly Notices of the Royal Astronomical Society*, vol. 345, pp. 1271–1284, Nov. 2003. 1.2

[9] B. C. Kelly, J. Bechtold, and A. Siemiginowska, "Are the Variations in Quasar Optical Flux Driven by Thermal Fluctuations?," *The Astrophysical Journal*, vol. 698, pp. 895–910, June 2009. 1.2.1

[10] Ž . Ivezić and C. MacLeod, "Optical variability of quasars: a damped random walk," *Proceedings of the International Astronomical Union*, vol. 9, pp. 395–398, oct 2013. 1.2.1

[11] S. Kozłowski, "Limitations on the recovery of the true AGN variability parameters using damped random walk modeling," *A&A*, vol. 597, p. A128, jan 2017. 1.2.1

[12] R. F. Mushotzky, R. Edelson, W. Baumgartner, and P. Gandhi, "Kepler observations of rapid optical variability in active galactic nuclei," *The Astrophysical Journal*, vol. 743, p. L12, nov 2011. 1.2.1

[13] V. P. Kasliwal, M. S. Vogeley, G. T. Richards, J. Williams, and M. T. Carini, "Do the kepler agn light curves need reprocessing?," *Monthly Notices of the Royal Astronomical Society*, vol. 453, pp. 2075–2081, aug 2015. 1.2.1

[14] T. Simm, M. Salvato, R. Saglia, G. Ponti, G. Lanzuisi, B. Trakhtenbrot, K. Nandra, and R. Bender, "Pan-STARRS1 variability of XMM-COSMOS AGN," *A&A*, vol. 585, p. A129, jan 2016. 1.2.1

[15] H. Guo, J. Wang, Z. Cai, and M. Sun, "How far is quasar UV/optical variability from a damped random walk at low frequency?," *The Astrophysical Journal*, vol. 847, p. 132, oct 2017. 1.2.1

[16] Y. Zu, C. S. Kochanek, S. Kozłowski, and A. Udalski, "IS QUASAR OPTICAL VARIABILITY a DAMPED RANDOM WALK?," *The Astrophysical Journal*, vol. 765, p. 106, feb 2013. 1.2.1

[17] V. P. Kasliwal, M. S. Vogeley, and G. T. Richards, "Extracting information from AGN variability," *Monthly Notices of the Royal Astronomical Society*, vol. 470, pp. 3027–3048, jun 2017. 1.2.1

[18] J. Moreno, M. S. Vogeley, G. T. Richards, and W. Yu, "Stochastic modeling handbook for optical AGN variability," *Publications of the Astronomical Society of the Pacific*, vol. 131, p. 063001, mar 2019. 1.2.1

[19] W. Yu, G. T. Richards, M. S. Vogeley, J. Moreno, and M. J. Graham, "Examining AGN UV/optical variability beyond the simple damped random walk," *The Astrophysical Journal*, vol. 936, p. 132, sep 2022. 1.2.1

[20] R. Vio and P. Andreani, "Comments on arxiv:1811.00154 [astro-ph.im] "agn variability analysis handbook"," 2018. 1.2.1.1

[21] B. M. Peterson, "The Broad-Line Region in Active Galactic Nuclei," in *Physics of Active Galactic Nuclei at all Scales* (D. Alloin, ed.), vol. 693, p. 77, 2006. 1.3

[22] D. Chelouche and E. Daniel, "Photometric Reverberation Mapping of the Broad Emission Line Region in Quasars," *The Astrophysical Journal*, vol. 747, p. 62, Mar. 2012. 1.3

[23] C. M. Gaskell and B. M. Peterson, "The Accuracy of Cross-Correlation Estimates of Quasar Emission-Line Region Sizes," *The Astrophysical Journal Supplement Series*, vol. 65, p. 1, Sept. 1987. 1.3

[24] R. A. Edelson and J. H. Krolik, "The Discrete Correlation Function: A New Method for Analyzing Unevenly Sampled Variability Data," *The Astrophysical Journal*, vol. 333, p. 646, Oct. 1988. 1.3

[25] T. Alexander, "Improved agn light curve analysis with the z-transformed discrete correlation function," 2013. 1.3.1

[26] T. Alexander, "ZDCF: Z-Transformed Discrete Correlation Function." Astrophysics Source Code Library, record ascl:1404.002, Apr. 2014. 1.3.1

[27] Y. Zu, C. S. Kochanek, and B. M. Peterson, "AN ALTERNATIVE APPROACH TO MEASURING REVERBERATION LAGS IN ACTIVE GALACTIC NUCLEI," *The Astrophysical Journal*, vol. 735, p. 80, jun 2011. 1.3.2

[28] Z. Yu, C. S. Kochanek, B. M. Peterson, Y. Zu, W. N. Brandt, E. M. Cackett, M. M. Fausnaugh, and I. M. McHardy, "On reverberation mapping lag uncertainties," *Monthly Notices of the Royal Astronomical Society*, vol. 491, pp. 6045–6064, Feb. 2020. 1.3.2, 4.4.4

[29] F. R. Donnan, K. Horne, and J. V. Hernández Santisteban, "Bayesian analysis of quasar light curves with a running optimal average: new time delay measurements of COSMOGRAIL gravitationally lensed quasars," *Monthly Notices of the Royal Astronomical Society*, vol. 508, pp. 5449–5467, Dec. 2021. 1.3.3, 3.2.3.2

[30] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, "emcee: The mcmc hammer," *Publications of the Astronomical Society of the Pacific*, vol. 125, pp. 306–312, mar 2013. 1.3.3

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997. 2, 5

[32] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. 2.1, 5

[33] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), p. 807–814, Omnipress, 2010. 2.1

[34] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010. 2.1

[35] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014. 2.1

[36] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013. 2.2, 5

[37] Y. LeCun and I. Misra, "Self-supervised learning: The dark matter of intelligence," Mar 2021. 2.2.1

[38] B. Naul, J. S. Bloom, F. Pérez, and S. van der Walt, "A recurrent neural network for classification of unevenly sampled variable stars," *Nature Astronomy*, vol. 2, pp. 151–155, nov 2017. 2.3, 3.2.2.2, 5

[39] Y. Tachibana, M. J. Graham, N. Kawai, S. G. Djorgovski, A. J. Drake, A. A. Mahabal, and D. Stern, "Deep modeling of quasar variability," *The Astrophysical Journal*, vol. 903, p. 54, nov 2020. 2.3, 3.2.2.2, 5

[40] P. Sá nchez-Sáez, H. Lira, L. Martí, N. Sánchez-Pi, J. Arredondo, F. E. Bauer, A. Bayo, G. Cabrera-Vives, C. Donoso-Oliva, P. A. Estévez, S. Eyheramendy, F. Förster, L. Hernández-García, A. M. M. Arancibia, M. Pérez-Carrasco, M. Sepúlveda, and J. R. Vergara, "Searching for changing-state AGNs in massive data sets. i. applying deep learning and anomaly-detection techniques to find AGNs with anomalous variability behaviors," *The Astronomical Journal*, vol. 162, p. 206, oct 2021. 2.3, 3.2.1, 3.2.2.2, 4.3.2, 4.3.2, 4.3.3, 5

[41] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008. 2.3

[42] C. Donoso-Oliva, I. Becker, P. Protopapas, G. Cabrera-Vives, M. Vishnu, and H. Vardhan, "ASTROMER," *A&A*, vol. 670, p. A54, feb 2023. 2.3, 5

[43] J. Pan, Y.-S. Ting, and J. Yu, "Astroconformer: Inferring surface gravity of stars from stellar light curves with transformer," 2022. 2.3, 5

[44] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," 2021. 2.3

[45] B. PENG, "Rwkv-lm," 2021. 2.3

[46] D. G. Koch, W. J. Borucki, G. Basri, N. M. Batalha, T. M. Brown, D. Caldwell, J. Christensen-Dalsgaard, W. D. Cochran, E. DeVore, E. W. Dunham, I. Gautier, Thomas N., J. C. Geary, R. L. Gilliland, A. Gould, J. Jenkins, Y. Kondo, D. W. Latham, J. J. Lissauer, G. Marcy, D. Monet, D. Sasselov, A. Boss, D. Brownlee, J. Caldwell, A. K. Dupree, S. B. Howell, H. Kjeldsen, S. Meibom, D. Morrison, T. Owen, H. Reitsema, J. Tarter, S. T. Bryson, J. L. Dotson, P. Gazis, M. R. Haas, J. Kolodziejczak, J. F. Rowe, J. E. Van Cleve, C. Allen, H. Chandrasekaran, B. D. Clarke, J. Li, E. V. Quintana, P. Tenenbaum, J. D. Twicken, and H. Wu, "Kepler Mission Design, Realized Photometric Performance, and Early Science," *The Astrophysical Journal*, vol. 713, pp. L79–L86, Apr. 2010. 2.3

[47] Y. Liu, P. Xiong, L. Xu, S. Cao, and Q. Jin, "Ts2-net: Token shift and selection transformer for text-video retrieval," 2022. 2.3

[48] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh, "Neural processes," 2018. 2.3

[49] J. Görtler, R. Kehlbeck, and O. Deussen, "A visual exploration of gaussian processes," *Distill*, 2019. https://distill.pub/2019/visual-exploration-gaussian-processes. 2.3

[50] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh, "Neural processes," 2018. 2.3, 5

[51] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, "Attentive neural processes," 2019. 2.3

[52] J. Gordon, W. P. Bruinsma, A. Y. K. Foong, J. Requeima, Y. Dubois, and R. E. Turner, "Convolutional conditional neural processes," 2020. 2.3

[53] A. Y. K. Foong, W. P. Bruinsma, J. Gordon, Y. Dubois, J. Requeima, and R. E. Turner, "Meta-learning stationary stochastic process prediction with convolutional neural processes," 2020. 2.3, 5

[54] Y. Dubois, J. Gordon, and A. Y. Foong, "Neural process family." `http://yanndubs.github.io/Neural-Process-Family/`, September 2020. 2.3

[55] I. Č vorović-Hajdinjak, A. B. Kovačević, D. Ilić, L. Č. Popović, X. Dai, I. Jankov, V. Radović, P. Sánchez-Sáez, and R. Nikutta, "Conditional neural process for nonparametric modeling of active galactic nuclei light curves," *Astronomische Nachrichten*, vol. 343, dec 2021. 2.3, 5

[56] J. W. Park, A. Villar, Y. Li, Y.-F. Jiang, S. Ho, J. Y.-Y. Lin, P. J. Marshall, and A. Roodman, "Inferring black hole properties from astronomical multivariate time series with bayesian attentive neural processes," 2021. 2.3, 5

[57] S. N. Shukla, *Deep Learning Models for Irregularly Sampled and Incomplete Time Series*. PhD thesis, 2021. 3.1

[58] S. N. Shukla and B. M. Marlin, "Multi-time attention networks for irregularly sampled time series," 2021. 3.1

[59] LSST Science Collaboration, P. A. Abell, J. Allison, S. F. Anderson, J. R. Andrew, J. R. P. Angel, L. Armus, D. Arnett, S. J. Asztalos, T. S. Axelrod, S. Bailey, D. R. Ballantyne, J. R. Bankert, W. A. Barkhouse, J. D. Barr, L. F. Barrientos, A. J. Barth, J. G. Bartlett, A. C. Becker, J. Becla, T. C. Beers, J. P. Bernstein, R. Biswas, M. R. Blanton, J. S. Bloom, J. J. Bochanski, P. Boeshaar, K. D. Borne, M. Bradac, W. N. Brandt, C. R. Bridge, M. E. Brown, R. J. Brunner, J. S. Bullock, A. J. Burgasser, J. H. Burge, D. L. Burke, P. A. Cargile, S. Chandrasekharan, G. Chartas, S. R. Chesley, Y.-H. Chu, D. Cinabro, M. W. Claire, C. F. Claver, D. Clowe, A. J. Connolly, K. H. Cook, J. Cooke, A. Cooray, K. R. Covey, C. S. Culliton, R. de Jong, W. H. de Vries, V. P. Debattista, F. Delgado, I. P. Dell'Antonio, S. Dhital, R. Di Stefano, M. Dickinson, B. Dilday, S. G. Djorgovski, G. Dobler, C. Donalek, G. Dubois-Felsmann, J. Durech, A. Eliasdottir, M. Eracleous, L. Eyer, E. E. Falco, X. Fan, C. D. Fassnacht, H. C. Ferguson, Y. R. Fernandez, B. D. Fields, D. Finkbeiner, E. E. Figueroa, D. B. Fox, H. Francke, J. S. Frank, J. Frieman, S. Fromenteau, M. Furqan, G. Galaz, A. Gal-Yam, P. Garnavich, E. Gawiser, J. Geary, P. Gee, R. R. Gibson, K. Gilmore, E. A. Grace, R. F. Green, W. J. Gressler, C. J. Grillmair, S. Habib, J. S. Haggerty, M. Hamuy, A. W. Harris, S. L. Hawley, A. F. Heavens, L. Hebb, T. J. Henry, E. Hileman, E. J. Hilton, K. Hoadley, J. B. Holberg, M. J. Holman, S. B. Howell, L. Infante, Z. Ivezic, S. H. Jacoby, B. Jain, R, Jedicke, M. J. Jee, J. Garrett Jernigan, S. W. Jha, K. V. Johnston, R. L. Jones, M. Juric, M. Kaasalainen, Styliani, Kafka, S. M. Kahn, N. A. Kaib, J. Kalirai, J. Kantor, M. M. Kasliwal, C. R. Keeton, R. Kessler, Z. Knezevic, A. Kowalski, V. L. Krabbendam, K. S. Krughoff, S. Kulkarni, S. Kuhlman, M. Lacy, S. Lepine, M. Liang, A. Lien, P. Lira, K. S. Long, S. Lorenz, J. M. Lotz, R. H. Lupton, J. Lutz, L. M. Macri, A. A. Mahabal, R. Mandelbaum, P. Marshall, M. May, P. M. McGehee, B. T. Meadows, A. Meert, A. Milani, C. J. Miller, M. Miller, D. Mills, D. Minniti, D. Monet, A. S. Mukadam, E. Nakar, D. R. Neill, J. A. Newman, S. Nikolaev, M. Nordby, P. O'Connor, M. Oguri, J. Oliver, S. S. Olivier, J. K. Olsen, K. Olsen, E. W. Olszewski, H. Oluseyi, N. D. Padilla, A. Parker, J. Pepper, J. R. Peterson, C. Petry, P. A. Pinto, J. L. Pizagno, B. Popescu, A. Prsa, V. Radcka, M. J. Raddick, A. Rasmussen, A. Rau, J. Rho, J. E. Rhoads, G. T. Richards, S. T. Ridgway, B. E. Robertson, R. Roskar, A. Saha, A. Sarajedini, E. Scannapieco, T. Schalk, R. Schindler, S. Schmidt, S. Schmidt, D. P. Schneider, G. Schumacher, R. Scranton, J. Sebag, L. G. Seppala, O. Shemmer, J. D. Simon, M. Sivertz, H. A. Smith, J. Allyn Smith, N. Smith, A. H. Spitz, A. Stanford, K. G. Stassun, J. Strader, M. A. Strauss, C. W. Stubbs, D. W. Sweeney, A. Szalay, P. Szkody, M. Takada, P. Thorman, D. E. Trilling, V. Trimble, A. Tyson, R. Van Berg, D. Vanden Berk, J. VanderPlas, L. Verde, B. Vrsnak, L. M. Walkowicz, B. D. Wandelt, S. Wang, Y. Wang, M. Warner, R. H. Wechsler, A. A. West, O. Wiecha, B. F. Williams, B. Willman, D. Wittman, S. C. Wolff, W. M. Wood-Vasey, P. Wozniak, P. Young, A. Zentner, and H. Zhan, "LSST Science Book, Version 2.0," *arXiv e-prints*, p. arXiv:0912.0201, Dec. 2009. 3.1

[60] C. Donoso-Oliva, I. Becker, P. Protopapas, G. Cabrera-Vives, V. M., and H. Vardhan, "Astromer: A transformer-based embedding for the representation of light curves," 2022. 3.1.2

[61] E. C. Bellm, S. R. Kulkarni, M. J. Graham, R. Dekany, R. M. Smith, R. Riddle, F. J. Masci, G. Helou, T. A. Prince, S. M. Adams, C. Barbarino, T. Barlow, J. Bauer, R. Beck, J. Belicki, R. Biswas, N. Blagorodnova, D. Bodewits, B. Bolin, V. Brinnel, T. Brooke, B. Bue, M. Bulla, R. Burruss, S. B. Cenko, C.-K. Chang, A. Connolly, M. Coughlin, J. Cromer, V. Cunningham, K. De, A. Delacroix, V. Desai, D. A. Duev, G. Eadie, T. L. Farnham, M. Feeney, U. Feindt, D. Flynn, A. Franckowiak, S. Frederick, C. Fremling, A. Gal-Yam, S. Gezari, M. Giomi, D. A. Goldstein, V. Z. Golkhou, A. Goobar, S. Groom, E. Hacopians, D. Hale, J. Henning, A. Y. Q. Ho, D. Hover, J. Howell, T. Hung, D. Huppenkothen, D. Imel, W.-H. Ip, Ž . Ivezić, E. Jackson, L. Jones, M. Juric, M. M. Kasliwal, S. Kaspi, S. Kaye, M. S. P. Kelley, M. Kowalski, E. Kramer, T. Kupfer, W. Landry, R. R. Laher, C.-D. Lee, H. W. Lin, Z.-Y. Lin, R. Lunnan, M. Giomi, A. Mahabal, P. Mao, A. A. Miller, S. Monkewitz, P. Murphy, C.-C. Ngeow, J. Nordin, P. Nugent, E. Ofek, M. T. Patterson, B. Penprase, M. Porter, L. Rauch, U. Rebbapragada, D. Reiley, M. Rigault, H. Rodriguez, J. van Roestel, B. Rusholme, J. van Santen, S. Schulze, D. L. Shupe, L. P. Singer, M. T. Soumagnac, R. Stein, J. Surace, J. Sollerman, P. Szkody, F. Taddia, S. Terek, A. V. Sistine, S. van Velzen, W. T. Vestrand, R. Walters, C. Ward, Q.-Z. Ye, P.-C. Yu, L. Yan, and J. Zolkower, "The zwicky transient facility: System overview, performance, and first results," *Publications of the Astronomical Society of the Pacific*, vol. 131, p. 018002, dec 2018. 3.2.1

[62] F. J. Masci, R. R. Laher, B. Rusholme, D. L. Shupe, S. Groom, J. Surace, E. Jackson, S. Monkewitz, R. Beck, D. Flynn, S. Terek, W. Landry, E. Hacopians, V. Desai, J. Howell, T. Brooke, D. Imel, S. Wachter, Q.-Z. Ye, H.-W. Lin, S. B. Cenko, V. Cunningham, U. Rebbapragada, B. Bue, A. A. Miller, A. Mahabal, E. C. Bellm, M. T. Patterson, M. Jurić , V. Z. Golkhou, E. O. Ofek, R. Walters, M. Graham, M. M. Kasliwal, R. G. Dekany, T. Kupfer, K. Burdge, C. B. Cannella, T. Barlow, A. V. Sistine, M. Giomi, C. Fremling, N. Blagorodnova, D. Levitan, R. Riddle, R. M. Smith, G. Helou, T. A. Prince, and S. R. Kulkarni, "The zwicky transient facility: Data processing, products, and archive," *Publications of the Astronomical Society of the Pacific*, vol. 131, p. 018003, dec 2018. 3.2.1

[63] M. J. Graham, S. R. Kulkarni, E. C. Bellm, S. M. Adams, C. Barbarino, N. Blagorodnova, D. Bodewits, B. Bolin, P. R. Brady, S. B. Cenko, C.-K. Chang, M. W. Coughlin, K. De, G. Eadie, T. L. Farnham, U. Feindt, A. Franckowiak, C. Fremling, S. Gezari, S. Ghosh, D. A. Goldstein, V. Z. Golkhou, A. Goobar, A. Y. Q. Ho, D. Huppenkothen, Ž . Ivezić, R. L. Jones, M. Juric, D. L. Kaplan, M. M. Kasliwal, M. S. P. Kelley, T. Kupfer, C.-D. Lee, H. W. Lin, R. Lunnan, A. A. Mahabal, A. A. Miller, C.-C. Ngeow, P. Nugent, E. O. Ofek, T. A. Prince, L. Rauch, J. van Roestel, S. Schulze, L. P. Singer, J. Sollerman, F. Taddia, L. Yan, Q.-Z. Ye, P.-C. Yu, T. Barlow, J. Bauer, R. Beck, J. Belicki, R. Biswas, V. Brinnel, T. Brooke, B. Bue, M. Bulla, R. Burruss, A. Connolly, J. Cromer, V. Cunningham, R. Dekany, A. Delacroix, V. Desai, D. A. Duev, M. Feeney, D. Flynn, S. Frederick, A. Gal-Yam,

M. Giomi, S. Groom, E. Hacopians, D. Hale, G. Helou, J. Henning, D. Hover, L. A. Hillenbrand, J. Howell, T. Hung, D. Imel, W.-H. Ip, E. Jackson, S. Kaspi, S. Kaye, M. Kowalski, E. Kramer, M. Kuhn, W. Landry, R. R. Laher, P. Mao, F. J. Masci, S. Monkewitz, P. Murphy, J. Nordin, M. T. Patterson, B. Penprase, M. Porter, U. Rebbapragada, D. Reiley, R. Riddle, M. Rigault, H. Rodriguez, B. Rusholme, J. van Santen, D. L. Shupe, R. M. Smith, M. T. Soumagnac, R. Stein, J. Surace, P. Szkody, S. Terek, A. V. Sistine, S. van Velzen, W. T. Vestrand, R. Walters, C. Ward, C. Zhang, and J. Zolkower, "The zwicky transient facility: Science objectives," *Publications of the Astronomical Society of the Pacific*, vol. 131, p. 078001, may 2019. 3.2.1

[64] S. Rakshit, C. S. Stalin, and J. Kotilainen, "Spectral Properties of Quasars from Sloan Digital Sky Survey Data Release 14: The Catalog," *The Astrophysical Journal Supplement Series*, vol. 249, p. 17, July 2020. 3.2.1

[65] M. J. Graham, S. G. Djorgovski, A. J. Drake, A. A. Mahabal, M. Chang, D. Stern, C. Donalek, and E. Glikman, "A novel variability-based method for quasar selection: evidence for a rest-frame ∼54 d characteristic time-scale," *Monthly Notices of the Royal Astronomical Society*, vol. 439, pp. 703–718, feb 2014. 3.2.1.1

[66] A. E. W. Johnson, T. J. Pollard, L. Shen, L.-W. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Sci. Data*, vol. 3, p. 160035, May 2016. 3.2.2.4

[67] S. N. Shukla and B. M. Marlin, "Interpolation-prediction networks for irregularly sampled time series," 2019. 3.2.2.4

[68] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, "Predicting in-hospital mortality of ICU patients: The PhysioNet/computing in cardiology challenge 2012," *Comput. Cardiol. (2010)*, vol. 39, pp. 245–248, 2012. 3.2.2.5

[69] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008. 3.2.3.1

[70] K. D. Denney, B. M. Peterson, R. W. Pogge, A. Adair, D. W. Atlee, K. Au-Yong, M. C. Bentz, J. C. Bird, D. J. Brokofsky, E. Chisholm, M. L. Comins, M. Dietrich, V. T. Doroshenko, J. D. Eastman, Y. S. Efimov, S. Ewald, S. Ferbey, C. M. Gaskell, C. H. Hedrick, K. Jackson, S. A. Klimanov, E. S. Klimek, A. K. Kruse, A. Ladéroute, J. B. Lamb, K. Leighly, T. Minezaki, S. V. Nazarov, C. A. Onken, E. A. Petersen, P. Peterson, S. Poindexter, Y. Sakata, K. J. Schlesinger, S. G. Sergeev, N. Skolski, L. Stieglitz, J. J. Tobin, C. Unterborn, M. Vestergaard, A. E. Watkins, L. C. Watson, and Y. Yoshii, "Reverberation Mapping Measurements of Black Hole Masses in Six Local Seyfert Galaxies," *The Astrophysical Journal*, vol. 721, pp. 715–737, Sept. 2010. 3.2.3.2

[71] Y.-R. Li, J.-M. Wang, L. C. Ho, P. Du, and J.-M. Bai, "A Bayesian Approach to Estimate the Size and Structure of the Broad-line Region in Active Galactic Nuclei Using Reverberation Mapping Data," *The Astrophysical Journal*, vol. 779, p. 110, Dec. 2013. 3.2.3.2

[72] M. M. Fausnaugh, D. A. Starkey, K. Horne, C. S. Kochanek, B. M. Peterson, M. C. Bentz, K. D. Denney, C. J. Grier, D. Grupe, R. W. Pogge, G. De Rosa, S. M. Adams, A. J. Barth, T. G. Beatty, A. Bhattacharjee, G. A. Borman, T. A. Boroson, M. C. Bottorff, J. E. Brown, J. S. Brown, M. S. Brotherton, C. T. Coker, S. M. Crawford, K. V. Croxall, S. Eftekharzadeh, M. Eracleous, M. D. Joner, C. B. Henderson, T. W. S. Holoien, T. Hutchison, S. Kaspi, S. Kim, A. L. King, M. Li, C. Lochhaas, Z. Ma, F. MacInnis, E. R. Manne-Nicholas, M. Mason, C. Montuori, A. Mosquera, D. Mudd, R. Musso, S. V. Nazarov, M. L. Nguyen, D. N. Okhmat, C. A. Onken, B. Ou-Yang, A. Pancoast, L. Pei, M. T. Penny, R. Poleski, S. Rafter, E. Romero-Colmenero, J. Runnoe, D. J. Sand, J. S. Schimoia, S. G. Sergeev, B. J. Shappee, G. V. Simonian, G. Somers, M. Spencer, D. J. Stevens, J. Tayar, T. Treu, S. Valenti, J. Van Saders, J. Villanueva, S., C. Villforth, Y. Weiss, H. Winkler, and W. Zhu, "Continuum Reverberation Mapping of the Accretion Disks in Two Seyfert 1 Galaxies," *The Astrophysical Journal*, vol. 854, p. 107, Feb. 2018. **??**, 4.4.1

[73] J. Thorne, "Reverberation mapping of the accretion discs in the quasars 3c 273 and 1h 2106-099," Master's thesis, Durham University, 2022. **??**, 4.4.3

[74] M. M. Fausnaugh, K. D. Denney, A. J. Barth, M. C. Bentz, M. C. Bottorff, M. T. Carini, K. V. Croxall, G. De Rosa, M. R. Goad, K. Horne, M. D. Joner, S. Kaspi, M. Kim, S. A. Klimanov, C. S. Kochanek, D. C. Leonard, H. Netzer, B. M. Peterson, K. Schnülle, S. G. Sergeev, M. Vestergaard, W. K. Zheng, Y. Zu, M. D. Anderson, P. Arévalo, C. Bazhaw, G. A. Borman, T. A. Boroson, W. N. Brandt, A. A. Breeveld, B. J. Brewer, E. M. Cackett, D. M. Crenshaw, E. Dalla Bontà, A. De Lorenzo-Cáceres, M. Dietrich, R. Edelson, N. V. Efimova, J. Ely, P. A. Evans, A. V. Filippenko, K. Flatland, N. Gehrels, S. Geier, J. M. Gelbord, L. Gonzalez, V. Gorjian, C. J. Grier, D. Grupe, P. B. Hall, S. Hicks, D. Horenstein, T. Hutchison, M. Im, J. J. Jensen, J. Jones, J. Kaastra, B. C. Kelly, J. A. Kennea, S. C. Kim, K. T. Korista, G. A. Kriss, J. C. Lee, P. Lira, F. MacInnis, E. R. Manne-Nicholas, S. Mathur, I. M. McHardy, C. Montouri, R. Musso, S. V. Nazarov, R. P. Norris, J. A. Nousek, D. N. Okhmat, A. Pancoast, I. Papadakis, J. R. Parks, L. Pei, R. W. Pogge, J. U. Pott, S. E. Rafter, H. W. Rix, D. A. Saylor, J. S. Schimoia, M. Siegel, M. Spencer, D. Starkey, H. I. Sung, K. G. Teems, T. Treu, C. S. Turner, P. Uttley, C. Villforth, Y. Weiss, J. H. Woo, H. Yan, and S. Young, "Space Telescope and Optical Reverberation Mapping Project. III. Optical Continuum Emission and Broadband Time Delays in NGC 5548," *The Astrophysical Journal*, vol. 821, p. 56, Apr. 2016. **??**, 4.4.2

[75] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. 4.1

[76] Y. Homayouni, J. R. Trump, C. J. Grier, Y. Shen, D. A. Starkey, W. N. Brandt, G. Fonseca Alvarez, P. B. Hall, K. Horne, K. Kinemuchi, J. I-Hsiu Li, I. D. McGreer, M. Sun, L. C. Ho, and D. P. Schneider, "The Sloan Digital Sky Survey Reverberation Mapping Project: Accretion Disk Sizes from Continuum Lags," *The Astrophysical Journal*, vol. 880, p. 126, Aug. 2019. 4.4.4

[77] R. Edelson, J. Gelbord, E. Cackett, B. M. Peterson, K. Horne, A. J. Barth, D. A. Starkey, M. Bentz, W. N. Brandt, M. Goad, M. Joner, K. Korista, H. Netzer, K. Page,

P. Uttley, S. Vaughan, A. Breeveld, S. B. Cenko, C. Done, P. Evans, M. Fausnaugh, G. Ferland, D. Gonzalez-Buitrago, J. Gropp, D. Grupe, J. Kaastra, J. Kennea, G. Kriss, S. Mathur, M. Mehdipour, D. Mudd, J. Nousek, T. Schmidt, M. Vestergaard, and C. Villforth, "The first swift intensive AGN accretion disk reverberation mapping survey," *The Astrophysical Journal*, vol. 870, p. 123, jan 2019. 4.4.4

[78] W. Yu and G. T. Richards, "EzTao: Easier CARMA Modeling." Astrophysics Source Code Library, record ascl:2201.001, Jan. 2022. 5

[79] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020. 5

APPENDIX A

Code Explanations and The Data Catalog

A.1 *DataSet* Class for Formatting the Light Curves

A nontrivial task of applying these models to account for and learn on incomplete and multivariate time series data is formatting the datasets to be ingested into the neural network. As such, we made a class to simplify and compartmentalize important aspects of this process and a utility function *get_data*() to build the dataset given access to some of the more dynamic preprocessing decisions. The main folder where the data exists should have subdirectories labeled by the band name with files in the band folders entitled such that the object reference or name should be sep arated by an underscore from the rest of the filename so the object's light curves can be matched properly to handle the multivariate case. The exemplar directory structure is shown in figure A.1.

Now we can walk through each successive method used in the body of *get_data*() A.2 to understand how the class is built and thus how the data is prepared. First we set the random seed, as is good practice for the sake of reproducibility in our shuffles or distribution samples. In any of the class methods that call permutations or sample from distributions, it is good practice to call the random seed again, such as *set_data_obj*()

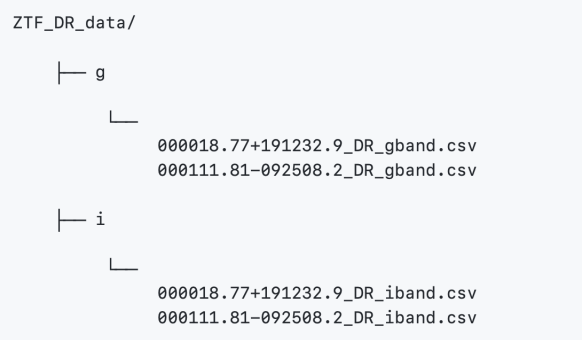


Figure A.1: Example directory structure for a dataset. In this case the prefix of the filenames are the SDSS Object IDs.

```
np.random.seed(seed=seed)
torch.manual_seed(seed=seed)
if folder.lower().find('ztf') > 0: lcs = ZtfDataSet(folder)
else: lcs = DataSet(folder)
lcs.files_to_df()
lcs.read(sep=sep)
lcs.prune(min_length=min_length, start_col=start_col,
          keep_missing=keep_missing)
if chop: lcs.chop_lcs()
lcs.resample_lcs(num_resamples=num_resamples, seed=seed)
lcs.normalize()
lcs.format_()
lcs.set_union_tp(uniform=True,n=n_union_tp)
print(f'dataset created, {lcs.dataset.shape=}')
lcs.set_data_obj(batch_size=batch_size, shuffle=shuffle,
                 test_split=test_split, seed=seed)
```

Figure A.2: Body of the *get_data* function.

or *resample_lcs*(). A dataset class is then initialized with the *folder* location, which calls the constructor in A.3. For clarity, all of the class attributes that are potentially set later on are initialized here.

```
def __init__(self, folder):
    if os.path.isdir(folder):
        self.folder = folder
    else:
        raise Exception(f"{folder} is not a directory")

    self.valid_files_df = pd.DataFrame() # keep track of the files across dimensions
    self.bands = []                      # array of band names
    self.dataset = []                    # array w/ragged dataset, format later on
    self.unnormalized_data = []          # keep unnormalized data for deprocessing
    self.union_tp = np.array([])         # union_tp for hetvae's intensity pathway
    self.target_x = np.array([])         # for making interpolations later on
    self.sigma_nxs = np.array([])        # normalized excess variance
    self.mean_mag = np.array([])         # mean magnitude
    self.med_cadence = np.array([])      # median cadence
```

Figure A.3: The constructor for the *DataSet* class.

After instantiating the object, we call *files_to_df*() in snippet A.4, which reads each band folder to successively match light curve files using a dataframe, so when parsing through each file in each band folder a new row is created for newly encountered objects with columns corresponding to the respective band and for already encountered objects, the dataframe is indexed by the object name. As stated initially, for this approach to work the object's name should be separated by an underscore from the rest of the filename and if no underscore is found, then the file without its extension is used. Being able to match the object or example names is irrelevant for univariate datasets, but for consistency's sake the files still must be placed in a band folder. The dataframe, *self.valid_files_df*, is filled with the object's file location and for missing light curves, filled with pandas' default (*nan*). We show an example of such a dataframe in figure A.5.

```python
def files_to_df(self):
    band_folders = [os.path.join(self.folder,bf) for bf in os.listdir(self.folder)]
    if band_folders and all([os.path.isdir(bf) for bf in band_folders]):
        for bf in band_folders:
            band = os.path.basename(bf)
            files = [os.path.join(bf, file) for file in os.listdir(bf)]
            print(f'found {len(files)} for {band=}')
            if len(files) > 0: self.bands.append(band)
            for file in files:
                if file.find('_') > 0:
                    obj_name = os.path.basename(file).split('_')[0]
                else:
                    # if no underscore to separate obj name, take filename as is
                    obj_name = os.path.splitext(os.path.basename(file))[0]
                self.valid_files_df.loc[obj_name, band] = file
    else:
        raise Exception('Empty data directory or files found where band folders should be.')
```

Figure A.4: *files_to_df* method.

	r	i	g
MCG+08-11-011	../datasets/ZTF_rm/r/MCG+08-11-011_DR_rband.csv	../datasets/ZTF_rm/i/MCG+08-11-011_DR_iband.csv	../datasets/ZTF_rm/g/MCG+08-11-011_DR_gband.csv
3C273	../datasets/ZTF_rm/r/3C273_DR_rband.csv	../datasets/ZTF_rm/i/3C273_DR_iband.csv	../datasets/ZTF_rm/g/3C273_DR_gband.csv
NGC2617	../datasets/ZTF_rm/r/NGC2617_DR_rband.csv	../datasets/ZTF_rm/i/NGC2617_DR_iband.csv	../datasets/ZTF_rm/g/NGC2617_DR_gband.csv
3C120	../datasets/ZTF_rm/r/3C120_DR_rband.csv	../datasets/ZTF_rm/i/3C120_DR_iband.csv	../datasets/ZTF_rm/g/3C120_DR_gband.csv
NGC5548	../datasets/ZTF_rm/r/NGC5548_DR_rband.csv	../datasets/ZTF_rm/i/NGC5548_DR_iband.csv	../datasets/ZTF_rm/g/NGC5548_DR_gband.csv
Mrk876	../datasets/ZTF_rm/r/Mrk876_DR_rband.csv	../datasets/ZTF_rm/i/Mrk876_DR_iband.csv	../datasets/ZTF_rm/g/Mrk876_DR_gband.csv
H2106-099	../datasets/ZTF_rm/r/H2106-099_DR_rband.csv	../datasets/ZTF_rm/i/H2106-099_DR_iband.csv	../datasets/ZTF_rm/g/H2106-099_DR_gband.csv
Mrk142	../datasets/ZTF_rm/r/Mrk142_DR_rband.csv	../datasets/ZTF_rm/i/Mrk142_DR_iband.csv	../datasets/ZTF_rm/g/Mrk142_DR_gband.csv
Mrk817	../datasets/ZTF_rm/r/Mrk817_DR_rband.csv	../datasets/ZTF_rm/i/Mrk817_DR_iband.csv	../datasets/ZTF_rm/g/Mrk817_DR_gband.csv

Figure A.5: Example of *valid_files_df*.

The next two class methods are *read*() and *prune*(). *read*() reads the files into a python array that is a list with lists for numpy arrays of the light curves for each object, with a placeholder array of zeros for every missing light curve in a given dimension, i.e. [[np.ndarray, np.ndarray, np.ndarray],[np.ndarray,np.ndarray,np.ndarray]] for three dimensions. It takes the parameter *sep*, which is the separator for the given data files with the default being a comma. The pruning method is the first case where its procedure would be specific to the dataset at hand, i.e. we want to filter out light curves close to the limiting magnitude of ZTF or we want to filter out ZTF error codes. To keep the relative generity of the filtering method, we can create a subclass of *DataSet* and override the functions that change with respect to the dataset.

The generic implementation of *prune*() aims to average duplicate points and filter out unworthy light curves and unworthy points in the light curves. It takes in four parameters: *start_col*, which dictates which column the observation times exist at, with mag/flux and magerr/fluxerr in two the columns following it, *min_length*, which is the minimum number of observations allowed for a valid light curve, *keep_missing*, the boolean value that decides whether or not we keep multivariate examples with missing light curves, and *std_threshold*, the number of standard deviations beyond which we would remove points as outliers. In the subclass' implementation of prune, ZTF specific filtering choices are included, like removing over-saturated light curves and those that are close to the limiting magnitude of ZTF based on the mean magnitude of the light curves or filtering ZTF-specific error codes relative to the 'catflags' column. The function is delicate in that the order of these processes matters and there are a lot of extra cases to account for. For instance, we have to account for the case that all the light curve's points might be removed when filtering outliers or the case that one round of duplicate averaging might create more duplicate points. We additionally attempt to let users know if *start_col* was set improperly, instead of letting the error propagate implicitly. We keep track of how many light curves are dropped per example because it all of the light curves are dropped for all the dimensions, then we can get rid of the entire example (in the files dataframe too). If *keep_missing* is set to false, we only keep examples that have light curves in every dimension; that is, none of their light curves have been replaced.

We now have a relatively clean, ragged array, still in the format like [[np.ndarray, np.ndarray, np.ndarray],[np.ndarray,np.ndarray,np.ndarray]]. Some accessory methods follow prune, like *chop_lcs*() and *resample_lcs*. Chopping the light curves might be necessary for HeTVAE if the distribution of lengths of the light curves is very right-skewed; that is, there are anomalously long lengths with respect to the larger proportion of the dataset. This is because of the way HeTVAE uses its reference time points. *resample_lcs* is a data augmentation tactic to add additional light curve examples as well as incorporate uncertainty information from the observational error by sampling from a distribution centered

```
def prune(self,start_col=1, min_length=1, keep_missing=True, std_threshold=10):
        replace = np.zeros((1,3)) # placeholder
        drops = []
        for i,object_lcs in enumerate(self.dataset):
            replace_count = 0
            for j,lc in enumerate(object_lcs):
                '''filter lcs w/ no points'''
                if not lc.any():
                    replace_count+=1
                    self.dataset[i][j] = replace
                    continue
                '''get columns for t,mag,magerr'''
                try:
                    lc = lc[:, start_col:start_col+3].astype(np.float32)
                    assert lc.shape[1] == 3
                except Exception:
                    raise Exception('double check start_col value')
                '''rm outliers'''
                y, y_mean, y_std = lc[:,1], np.mean(lc[:,1]), np.std(lc[:,1])
                outliers = np.where((y > (y_mean + y_std*std_threshold)) | \
                                    (y < (y_mean - y_std*std_threshold)))[0]
                lc = np.delete(lc, outliers, axis=0)
                if len(lc) == 0:
                    replace_count+=1
                    self.dataset[i][j] = replace
                    continue
                '''average duplicate points'''
                while len(lc) != len(np.unique(lc[:,0])):
                    lc = np.array([lc[np.where(i==lc[:,0])[0]].mean(0) for i in np.
                        unique(lc[:,0])])
                '''min length filter'''
                if len(lc) < min_length:
                    replace_count+=1
                    self.dataset[i][j] = replace
                    continue
                self.dataset[i][j] = lc
            if (replace_count >= len(self.bands)) or \
            (not keep_missing and replace_count > 0):
                drops.append(i)
        self.valid_files_df.drop(self.valid_files_df.index[drops], inplace=True)
        self.dataset = [self.dataset[i] for i in range(len(self.dataset)) if i not in
            drops]
```

Figure A.6: Code for *prune*().

at zero with a standard deviation equal to the observational error for each observation and adding that sample to the original mag/flux value.

The normalization method is next and its code is self-explanatory, but we also keep a copy of the unnormalized data in $unnormalized\_data$ for future de-normalization purposes. Apart from $prune()$, the most hard-working function is $format\_()$, that fixes the ragged list of lists of numpy arrays into one proper numpy array. To do this time values must be the same for all the light curves corresponding to an object or for one multivariate example. As such, the intra-example time values are combined and magnitude/flux values are placed according to the newly unified time array with zeroes where there are not recorded observations. Then, the longest of the intra-example union time arrays across the entire dataset informs us of how many zeroes we need to fill other examples with to make all of the example's lengths uniform.

```
Intra-example formatting:

light curve g

  t    mag  magerr
[[0.   4.3  0.09]
 [1.2  5.2  0.08]]

light curve r

[[0.   3.2  0.12]
 [2.33 4.1  0.03]]
```

```
becomes:

light curve g

[[0.   4.3  0.09]
 [1.2  5.2  0.08]
 [2.33 0.   0.  ]]

light curve r

[[0.   3.2  0.12]
 [1.2  0.   0.  ]
 [2.33 4.1  0.03]]
```

```
if 5 is the length of the longest
intra-example union-time array, becomes:

light curve g

[[0.   4.3  0.09]
 [1.2  5.2  0.08]
 [2.33 0.   0.  ]
 [0.   0.   0.  ]
 [0.   0.   0.  ]]

light curve r

[[0.   3.2  0.12]
 [1.2  0.   0.  ]
 [2.33 4.1  0.03]
 [0.   0.   0.  ]
 [0.   0.   0.  ]]
```

Figure A.7: Downsized example of intra-example light curve formatting.

$set\_union\_tp()$ is very important because of its relevance to the *intensity* pathway of HeTVAE. Its parameters, a boolean, $uniform$, if set to $True$, and n decide if the array is to be evenly spaced, with n points from 0 to the maximum time value across all light curves. If uniform is $False$, then $self.union\_tp$ will be the union of all the time points across the dataset.

$set\_data\_obj()$ is the final stage of data preparation so that the dataset can be digested

```
def format_(self):
        union_tp_ex = [np.unique(np.hstack([lc[:,0] for lc in object_lcs])) for
            object_lcs in self.dataset]
        union_tp_max = np.max([len(utp) for utp in union_tp_ex]) # to zero fill w/
        for i, object_lcs in enumerate(self.dataset):
            for j, lc in enumerate(object_lcs):
                # need to reformat the observations relative to union_tp
                new_y = np.zeros_like(union_tp_ex[i])
                new_yerr = np.zeros_like(union_tp_ex[i])
                mask = np.isin(union_tp_ex[i], lc[:,0]) # where are the timepoints
                    relative to union_
                indexes = np.nonzero(mask)[0]
                new_y[indexes] = lc[:,1]
                new_yerr[indexes] = lc[:,2]
                # set the new time series that is correctly formatted
                new_lc = np.array([union_tp_ex[i], new_y, new_yerr]).T
                new_lc = np.append(new_lc, np.zeros(((union_tp_max - new_lc.shape[0]),3)
                    ),axis=0)
                self.dataset[i][j] = new_lc
        self.dataset = np.array(self.dataset,dtype=np.float32)
        print(self.dataset.shape)
```

Figure A.8: Code for *format*().

```
def set_union_tp(self, uniform=False, n=1000):
        """
        calcluates an array of the union of all the time points across the dataset
            unless uniform==True,
        in that case set union_tp to n uniformly spaced points between the maximum and
            minimum observed time
        across the dataset
        args:
            uniform       (boolean)   --> uniformly spread?
            n             (int)       --> number of uniformly spread pts
        side effects
            - sets self.union_tp
            - prints length of union tp
        """
        self.union_tp = np.unique(self.dataset[:,:,:,0].flatten())
        print('max time: ', np.max(self.union_tp))
        if uniform:
            step = np.ptp(self.union_tp) / n
            self.union_tp = np.arange(np.min(self.union_tp), np.max(self.union_tp), step
                )
        self.union_tp = self.union_tp.astype(np.float32)
        print(f'created union_tp attribute of length {len(self.union_tp)}')
```

Figure A.9: Code for *set_union_tp*().

```
def set_data_obj(self, batch_size=8, test_split=0.1, shuffle=False, seed=2):
        if shuffle: # keep a consistent shuffle for unprocessing
            np.random.seed(seed=seed)
            torch.manual_seed(seed=seed)
            shuf = np.random.permutation(len(self.dataset))
            self.dataset = self.dataset[shuf]
            self.unnormalized_data = [self.unnormalized_data[i] for i in shuf]
            self.valid_files_df = self.valid_files_df.reindex(self.valid_files_df.index[
                shuf])
        # val and train set can be the same because light curves are conditioned
        # on differing subsamples
        splindex = int(np.floor((1-test_split)*len(self.dataset)))
        training, test = np.split(self.dataset, [splindex])
        valid_splindex = int(np.floor(0.8*len(training)))
        _, valid = np.split(training, [valid_splindex])
        self.test_split_index = splindex # keep this for denormalizing later
        print(f'train size: {len(training)}, valid size: {len(valid)}, test size: {len(
            test)}')
        train_loader = torch.utils.data.DataLoader(training, batch_size=batch_size)
        valid_loader = torch.utils.data.DataLoader(valid, batch_size=batch_size)
        test_loader = torch.utils.data.DataLoader(test, batch_size=batch_size)
        union_tp = torch.Tensor(self.union_tp)

        self.data_obj = {
            "train_loader": train_loader,
            "test_loader": test_loader,
            "valid_loader": valid_loader,
            'union_tp': union_tp,
            "input_dim": len(self.bands),
        }
```

Figure A.10: Code for *set_data_obj*().

by the neural network. The first parameter, *batch_size*, sets the size of the batch for the PyTorch dataloader. The second, *test_split*, is the proportion of the training data to split to the test set. The third, *shuffle*, is a boolean that decides whether or not to shuffle the dataset, ensuring that the files dataframe and the unnormalized dataset is shuffled the same way for de-processing and latent analysis. Finally, because we have a shuffle happening, the random *seed* parameter is necessary.

The entirety of the class can be found at `https://github.com/mwl10/hetast/blob/master/src/dataset.py` as we omit some of the code for brevity.

A.2 Data Catalog

The catolog used for training can be found here:

https://htmlpreview.github.io/?https:/github.com/mwl10/hetast/blob/master/....