STACCATO: A Novel Solution to Supernova Photometric Classification with Biased Training sets

Esben A. Revsbech[1], R. Trotta[2,3,4⋆] and David A. van Dyk[1,3,4]
[1] *Statistics Section, Mathematics Department, Huxley Building, South Kensington Campus, Imperial College London, London SW7 2AZ*
[2] *Astrophysics Group, Physics Department, Imperial College London, Prince Consort Rd, London SW7 2AZ*
[3] *Imperial Centre for Inference and Cosmology, Astrophysics Group, Blackett Laboratory, Prince Consort Rd, London SW7 2AZ*
[4] *Data Science Institute, William Penney Laboratory, Imperial College London, London SW7 2AZ*

27th September 2017

ABSTRACT

We present a new solution to the problem of classifying Type Ia supernovae from their light curves alone given a spectroscopically confirmed but biased training set, circumventing the need to obtain an observationally expensive unbiased training set. We use Gaussian processes (GPs) to model the supernovae's (SN) light curves, and demonstrate that the choice of covariance function has only a small influence on the GPs ability to accurately classify SNe. We extend and improve the approach of Richards et al. (2012) — a diffusion map combined with a random forest classifier — to deal specifically with the case of biassed training sets. We propose a novel method, called STACCATO ('*S*yn*t*hetically *A*ugmented Light *C*urve *C*lassific*ation*') that synthetically augments a biased training set by generating additional training data from the fitted GPs. Key to the success of the method is the partitioning of the observations into subgroups based on their propensity score of being included in the training set. Using simulated light curve data, we show that STACCATO increases performance, as measured by the area under the Receiver Operating Characteristic curve (AUC), from 0.93 to 0.96, close to the AUC of 0.977 obtained using the 'gold standard' of an unbiased training set and significantly improving on the previous best result of 0.88. STACCATO also increases the true positive rate for SNIa classification by up to a factor of 50 for high-redshift/low brightness SNe.

Key words: Supernovae Type Ia, Bayesian statistics, cosmological parameters, classification

1 INTRODUCTION

Supernovae Type Ia (SNIa) have been crucial in establishing the accelerated expansion of the Universe (Perlmutter et al. 1999; Riess et al. 1998). They are expected to remain important distance indicators in the next few years, as the worldwide sample of SNIa is set to increase many fold thanks to ongoing and upcoming observational programmes. The Dark Energy Survey (DES) is expected to observe $\sim 3,000$ SNIa over 5 years, while the Large Synoptic Survery Telescope (LSST) is expected to observe of the order of $\sim 10^5$ SNIa each year (LSST Science Collaboration et al. 2009).

One of the major bottlenecks for the cosmological analysis of such a large and rich data set is the classification of SNIa candidates. Traditionally, SNIa candidates have been confirmed by spectroscopic follow-up, as SNIa are characterized by the lack of H in their spectrum and the presence of strong SiII lines. However, spectroscopic follow up of thousands of candidates is simply not observationally feasible. Thus it is becoming crucial to reliably classify SNIa on the basis of photometric information alone (Kessler et al. 2010a). In parallel, methods are being developed to fit cosmological models to a SN sample contaminated by non-type Ia (Kunz et al. 2007; Hlozek et al. 2012; Knights et al. 2013; Kessler & Scolnic 2017), which generally require as input the probability for a given object to be of type Ia (based on photometry alone).

To address this problem, Kessler et al. (2010a) set up a "Supernova photometric classification challenge", inviting teams to develop methods for SNIa classification from a suite of numerical simulations designed to mock data from DES. The simulations contain SN Type Ia, Ib, Ic and II light curves (LCs) with realistic noise and selection effects and are divided into a training set and a testing set. SN types are known in the training set so that it can be used to tune the classifier. During the challenge, SN types for the test set

⋆ Corresponding author: r.trotta@imperial.ac.uk

were not revealed until completion of the challenge so that this set could be used to evaluate the performance of the proposed classifiers. Teams were evaluated on a Figure of Merit (FoM) that combines both the efficiency and the purity of the classification of the test set. There were 13 entries from 10 teams, using a variety of strategies. The broad conclusions from the original challenge are that none of the methods adopted was clearly superior, and that an overall major difficulty is the fact that the realistic training set was not representative of the test set. This is a consequence of the fact that the observer-frame magnitude cut used to define the training set map onto different brightness levels as a function of redshift for SNIa and core collapse SNe (as their rest-frame spectrum is different). As a consequence, the ratio of SNIa to non-Ia SNe is different for the training and the test set, as they have different magnitude cuts, with the proportion of non-Ia SNe being generally underrepresented at high redshift/low apparent brightness. Moreover, there are very few high redshift or low apparent magnitude SNe *of any type* in the training set. Unfortunately, tuning a classifier with such an unrepresentative training set leads to poor results, especially for high redshift and/or dim SNe. Newling et al. (2011); Varughese et al. (2015); Lochner et al. (2016) found considerable decreases in performance if the training set is biased. Richards et al. (2012) suggested and evaluated several strategies for choosing additional SNIa for spectroscopic follow-up. Lochner et al. (2016) carried out a comparison of five different methods and found that the classification of all of them degraded significantly when using a biased training set (of the kind that is likely to be available in practice) as opposed to a representative sample. They concluded that a representative training set is "essential" for good classification when using these methods.

Accurate classification based on a biased training set is generally problematic. In this work we build on the methods of Richards et al. (2012) and Lochner et al. (2016), and suggest a novel approach to SNIa photometric classification that only uses a biased training set. We name this classification approach, '*Syn*thetically *A*ugmented Light *C*urve *C*lassific*ation*' (STACCATO)[1]. In our approach, the effects of the bias are mitigated by dividing the training set into a number of groups based on the propensity score, i.e. the probability of belonging to the training set given a number of covariates (here, redshift and apparent brightness). Because some of these groups have very small training sets, we propose to augment the training set by sampling new synthetic LCs from Gaussian Processes (GPs) fit to the original training set LCs. We show that this strategy improves the classification accuracy considerably, to a level that compares favourably with the performance obtained from a unbiased training set. In the current implementation the choice of degree of augmentation of the training sets by synthetic LCs involves an optimization step that requires knowledge of the SNe types in a subset of the test set. We leave to future work designing and demonstrating a procedure that does not require such data, which would of course not be available in a real situation. At the end of this paper, however, we are able to demonstrate that even without the optimization step, our augmentation strategy performs better than using the original training set without augmentation.

Our overall strategy is to use GPs to build probabilistic models for the individual LCs. We then carry out a classification step in which a diffusion map and a random forest classifier are applied. Here we draw on the work of Richards et al. (2012), but we use a different specification of the LCs metric, and compute the diffusion map separately for each filter and only on the relevant training data. In the final, entirely novel step, we apply a propensity score to the training set, which we then augment with synthetic LCs generated probabilistically from the fitted GPs. We demonstrate that this approach circumvents the need to design an observationally expensive unbiased training set, and that the performance of the classifier (as measured by the Area under the ROC Curve, AUC) is improved from 0.92 to 0.96 (as compared to 0.977 for an unbiased training set), with most of the improvement coming from the highest redshift SN group (with the fewest SNe in its training set). This compares favourably with the best methods in Lochner et al. (2016), which delivered about 0.88 for the biased training set.

The structure of this paper is as follows. In Section 2 we describe our Gaussian Process fit to the LC data; in Section 3 we describe a normalization and time-alignment procedure for the LCs; in Section 4 we explain how we classify SN with a diffusion map coupled with a random forest classifier; in Section 5 we present the classification results and contrast the case of a biased and an unbiased training set; in Section 6 we present the STACCATO approach and discuss the improvements it brings to the classification result; finally, we conclude in Section 7.

2 GAUSSIAN PROCESS LIGHT CURVE FIT

2.1 Light Curves Data and Training Sets

We use the dataset that was originally released by Kessler et al. (2010a) as part of their "Supernova photometric classification challenge". Since the release of that dataset, an updated version has been made available (Kessler et al. 2010b), with various improvements and a few bug fixes. In order to compare our results with the outcome of the original challenge, we used exactly the same dataset that was available to the teams that entered the challenge. More recent works have used the updated dataset, and we will apply our methodology to this newer and more realistic dataset in a future paper.

The dataset from Kessler et al. (2010a) contains photometric LCs from 18,321 simulated SNe. For each SN, LCs are given in four colour bands, $C = (g, r, i, z)$, measured at unevenly spaced points in time. The observations are subject to measurement error and estimated standard deviations are given as part of the dataset. The number of observations times for the LCs ranges from 0 to 41 with a mean of 19.1 and a median of 21. We only use SNe with at least three observations in each band, thus reducing the dataset to 17,330 SNe. The challenge included both a set with host galaxy photometric redshift, and one without it. In this paper, we only analyze the data set with host galaxy redshift. Lochner

[1] An R code implementing STACCATO and allowing the user to reproduce all the results and figures in this paper is available from: `https://github.com/rtrotta/STACCATO`.

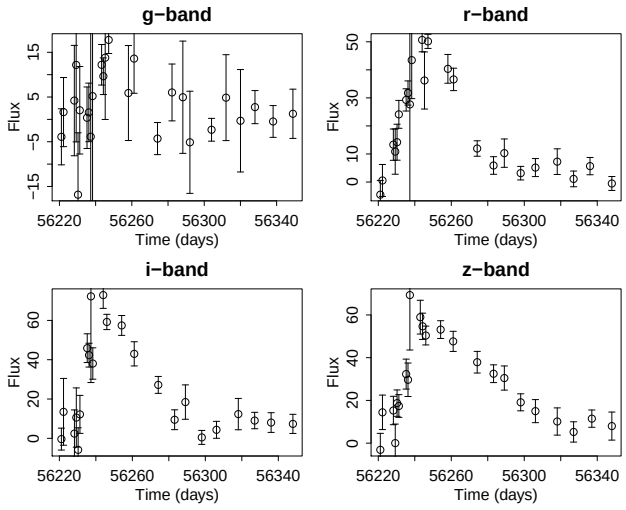

Figure 1. An example of LC data in four bands for (the randomly selected) SN194156 at $z = 0.54$. Vertical 1σ error bars are also plotted.

et al. (2016) found that removing redshift information did not significantly degrade the classification performance. We will explore this aspect in future work.

Figure 1 displays the four LCs of a randomly chosen SN (at redshift 0.54) and illustrates both the very noisy behaviour (in the g band) that some SNe exhibit in one or more bands and the more 'well behaved' peak structure that is typically associated with SNIa explosions.

The simulated dataset from Kessler et al. (2010a) is divided into a training set, $\mathcal{B}_{\text{train}}$, of 1,217 SNe with known types and a test set, $\mathcal{B}_{\text{test}}$, of 16,113 simulated SNe with unknown types. $\mathcal{B}_{\text{train}}$ is obtained by simulating the spectroscopic follow-up efficiency from a 4m class telescope with a limiting r-band magnitude of 21.5, and an 8m class telescope with a limiting i-band magnitude of 23.5. SNIa LCs are simulated from a mix of SALT-II and MLCS models, while non-SNIa are simulated from a set of 41 templates derived from spectroscopically confirmed non-SNIa (Kessler et al. 2010b). The goal is to use $\mathcal{B}_{\text{test}}$ to classify the SN in $\mathcal{B}_{\text{test}}$. Due to observational selection effects, the spectroscopic training set is biased in terms of SN types, redshift, and brightness. This bias is mimicked in the dataset of Kessler et al. (2010a) so that there are proportionally more bright, low redshift SNIa in $\mathcal{B}_{\text{train}}$ than in $\mathcal{B}_{\text{test}}$.

We also construct an *unbiased* training set, $\mathcal{U}_{\text{train}}$, by random sampling 1,200 SNe from the entire dataset. Here we exploit the fact that the classes of the entire dataset was released post challenge. The remaining data is assigned to a corresponding test set, $\mathcal{U}_{\text{test}}$, used for evaluating the performance of the classifier. For consistency, the sizes of $\mathcal{U}_{\text{train}}$ and $\mathcal{B}_{\text{train}}$ are similar. We refer to the $\mathcal{U}_{\text{train}}$ as 'the gold standard', as it is a 'best case scenario' to compare any classification algorithm against. Although, such an unbiased training set is not feasible in practice, we want to assess the reduction in the classifier performance that can be attributed to the bias in $\mathcal{B}_{\text{train}}$. The composition of both training and test sets is summarized in Table 1.

2.2 Modelling Light Curves with Gaussian Processes

Let $X(t)$ be a stochastic process with continuous time index, t, in some time interval, T. We say $X(t)$ follows a Gaussian Process (GP) (e.g., Adler 1990), if the finite dimensional distribution, $p(X(t_1), \ldots, X(t_k))$, is (multivariate) Gaussian for any positive integer, k, and any set of time points $t_1, \ldots, t_k$ in T. Two key theoretical results are the existence and uniqueness of the Gaussian process. Specifically, a GP is uniquely determined by its mean function,

$$\mu(t) = \mathrm{E}[X(t)] \tag{1}$$

and its covariance function,

$$K(t,s) = \mathrm{E}\left[\{X(t) - \mu(t)\}^T \{X(s) - \mu(s)\}\right], \tag{2}$$

where t and s are any two time points in T. Conversely for any given mean and covariance functions, there exists a GP with these mean and covariance functions. (For previous applications of GP regression to SN LC fitting, see Kim et al. (2013).)

The key result that allows us to use GPs to model time series such as LCs stems from the conditioning rule for multivariate Gaussian distributions (e.g., Rasmussen & Williams 2006). Suppose, for example, that X follows a multivariate Gaussian distribution with mean vector m and variance matrix Σ, i.e., $X \sim N(m, \Sigma)$, and partition

$$X = \begin{Bmatrix} X_1 \\ X_2 \end{Bmatrix}, \; m = \begin{Bmatrix} m_1 \\ m_2 \end{Bmatrix}, \text{ and } \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}.$$

The conditional distribution of X_2 given X_1 is also a (multivariate) Gaussian, specifically $X_2 \mid X_1 \sim N(m_*, \Sigma_*)$ with

$$\begin{aligned} m_* &= \mathrm{E}[X_2|X_1] = m_2 + \Sigma_{21}\Sigma_{11}^{-1}(X_1 - m_1) \\ \Sigma_* &= \mathrm{Var}(X_2|X_1) = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}. \end{aligned} \tag{3}$$

Turning to the modeling of LCs, let $f(t)$ denote an unobserved SN LC continuous in time. Suppose that

$$f \sim GP(\mu, K), \tag{4}$$

where $GP(\mu, K)$ denotes a GP with mean and covariance functions μ and K. (Here and elsewhere we suppress the dependence of f, μ, and K on time.) In practice, we must specify the functional forms of μ and K, typically in terms of several unknown parameters. For the moment, we assume μ and K are given, putting off discussion of their specification until Section 2.3.

Because the distribution of $f(t)$ at any finite set of time points is multivariate Gaussian, given a series of observations we can simply apply the formulas in (3) to obtain the conditional distribution of $f(t)$ at any other finite set of time points given the observed values. In this way, we can interpolate $f(t)$ between the observed values. Specifically, if we measure at n points in time a vector of observations $f_{\text{obs}} = (f(t_1), \ldots, f(t_n))$, we can obtain the conditional distribution of $f(t)$ at another set of k time points, namely $\tilde{f} = (f(\tilde{t}_1), \ldots, f(\tilde{t}_k))$, by conditioning on the observations,

$$\tilde{f} \mid f_{\text{obs}} = \begin{pmatrix} f(\tilde{t}_1) \\ \vdots \\ f(\tilde{t}_k) \end{pmatrix} \Bigg| \begin{pmatrix} f(t_1) \\ \vdots \\ f(t_n) \end{pmatrix} \sim N_k\left(m_*, \Sigma_*\right), \tag{5}$$

	Type Ia		Type II		Type Ibc		Total
$\mathcal{B}_{\text{train}}$	851	(69.9%)	257	(21.2%)	109	(9.0%)	1,217
$\mathcal{U}_{\text{train}}$	292	(24.3%)	749	(62.4%)	159	(13.2%)	1,200
$\mathcal{B}_{\text{test}}$	3,592	(22.3%)	10,481	(65.0%)	2,040	(12.6%)	16,113
$\mathcal{U}_{\text{test}}$	4,151	(25.7%)	9,989	(61.9%)	1,990	(12.3%)	16,130

Table 1. Composition of training and testing datasets. $\mathcal{B}_{\text{test}}$ is a realistic biased training set, while $\mathcal{U}_{\text{train}}$ is the 'gold standard' unbiased training set. We compare the performance of our algorithm using both test sets to assess the effect of the biased training set on the classification quality.

where m_* and Σ_* are in (3) with $m_1 = (\mu(t_1), \ldots, \mu(t_n))^T$, $m_2 = \left(\mu(\tilde{t}_1), \ldots, \mu(\tilde{t}_k)\right)^T$, $\Sigma_{11} = \mathbf{K}(t,t)$, $\Sigma_{12} = \mathbf{K}(t,\tilde{t})$, $\Sigma_{21} = \Sigma_{12}^T$, and $\Sigma_{22} = \mathbf{K}(\tilde{t},\tilde{t})$, where $\mathbf{K}(t,\tilde{t})$ is a matrix with element (i,j) equal to $K(t_i,\tilde{t}_j)$. In Bayesian terms (4) is the prior distribution for f and (5) is the posterior distribution of $f(t)$ evaluated at a set of unobserved times. Thus, we refer to (4) as the prior GP, to (5) as the posterior GP, and to the vector m_* as the posterior mean function. We use the posterior mean functions as the fitted LCs in our classification of SNe.

Although the derivation of the posterior distribution in (5) does not account for measurement errors, they can easily be included in an elegant manner. This is a prime motivation for the use of GPs to model LCs. Assuming uncorrelated Gaussian errors, let y_i and σ_i denote the observed flux and standard error at time t_i, for $i = 1, \ldots, n$. The data can be modeled as

$$y_i = f(t_i) + \sigma_i \varepsilon_i, \tag{6}$$

where $f \sim GP(\mu, K)$ and ε_i is iid $N(0,1)$ for $i = 1, \ldots, n$. Thus, the posterior distribution of f at a given set of k time points $\tilde{t}_1, \ldots, \tilde{t}_k$ is simply

$$\tilde{f}|y \sim N_k(m_*, \Sigma_*), \tag{7}$$

where $\tilde{f} = f(\tilde{t})$ and m_* and Σ_* are as in (5) except that the noise vector $\sigma^2 = (\sigma_1^2, \ldots, \sigma_n^2)^T$ is added as a diagonal matrix to Σ_{11} so $\Sigma_{11} = \mathbf{K}(t,t) + \text{diag}(\sigma^2)$.

2.3 Mean and Covariance Functions

The choice of μ and K can have a large influence on the quality of the LC interpolation in (5). The covariance function, for example, controls the possible LC 'shapes' that can be generated under the prior GP and hence depending on the amount and quality of data may influence the posterior GP used for interpolation, namely the multivariate Gaussian distribution in (5).

In regions with abundant data the posterior GP is dominated by the observations, and the mean function has little influence. Therefore, the main focus of the literature is the covariance function, and the mean function is often simply taken to be a constant equal to zero, $\mu(t) = 0$. We adopt this choice and although the LC data often include sparse regions, we observe only limited problems with the posterior GP drifting towards the prior mean function, i.e., zero.

There is a wide range of standard choices for covariance functions, each designed to model specific features of the data. These can be summed or multiplied (Rasmussen & Williams 2006) to obtain a large and flexible strategy for specifying covariance structure (see e.g., Gelman et al. 2013). Because data for the individual SN LCs are limited, such refined models are not appropriate. Instead we consider two relatively simple and flexible covariance functions, namely the squared exponential and the Gibbs kernels.

2.3.1 *Squared Exponential Kernel*

The squared exponential (SE) kernel is a popular choice (Roberts et al. 2012) and has been used before in fitting SN LCs (Lochner et al. 2016; Kim et al. 2013). It is given by

$$K_{\text{se}}(t,s) = \tau^2 \exp\left(-\frac{1}{2}\frac{(t-s)^2}{l^2}\right), \tag{8}$$

where τ^2 and l are free parameters. The parameter l is the length scale parameter and controls the speed with which the covariance decreases over time, i.e., how quickly the GP 'forgets' previous observations and thus how rapidly it can fluctuate. The parameter τ^2 is the variance parameter and controls the scale of the covariance function and hence the 'amplitude' of the process. Notice that $K_{\text{se}}(t,s)$ depends on (t,s) only through the difference $|t-s|$, and hence corresponds to a stationary process. The kernel is continuous and infinitely differentiable with respect to $r = t - s$. This means that the prior GP is smooth in the mean square sense, and we can expect the posterior GP and mean function to be smooth as well.

2.3.2 *Gibbs Kernel*

The Gibbs kernel can be viewed as a generalization of the SE kernel in which the length scale parameter, l, is allowed to vary over time. This means that the timescale over which the GP remembers past observations and thus the timescale for variability can change. The added flexibility might be useful in accommodating different degrees of smoothing in different parts of the time domain, e.g., due to different levels of sparsity in the data. Specifically, the Gibbs (1997) kernel is given by,

$$K_{\text{Gibbs}}(t,s) = \tau^2 \left(\frac{2l(t;\theta)l(s;\theta)}{l^2(t;\theta) + l^2(s;\theta)}\right)^{1/2} \exp\left(-\frac{(t-s)^2}{l^2(t;\theta) + l^2(s;\theta)}\right), \tag{9}$$

where $l(t;\theta)$ is a positive length scale function depending on time and a multivariate parameter θ. We set

$$l(t;\theta) = \lambda\left(1 - p\,\varphi_{(t_{\max},\eta)}(t)\right), \tag{10}$$

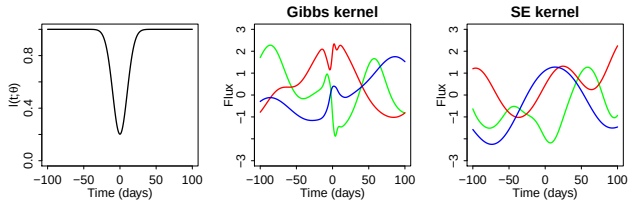

Figure 2. *Left*: The length scale function for the Gibbs kernel with $\lambda = 1$, $p = 20$, $t_{\max} = 0$ and $\eta = 10$. *Middle*: Three functions drawn from the GP prior, $\mathrm{GP}(0, K_{\mathrm{Gibbs}})$, with the Gibbs kernel and length scale function in the left plot and with $\lambda = 20$ (green), $\lambda = 30$ (red), and $\lambda = 40$ (blue). *Right*: Three functions drawn from $\mathrm{GP}(0, K_{\mathrm{se}})$ with $\tau = 1$ and $l = 20$ (green), $l = 30$ (red), and $l = 40$ (blue).

where λ, p, $t_{\max}$, and η are tuning parameters and $\varphi_{(t_{\max},\eta)}(t)$ is a Gaussian density with mean $t_{\max}$ and standard deviation η. With $p = 0$ the Gibbs kernel reverts to the SE kernel. For $t = s$, $K_{\mathrm{Gibbs}}(t, t) = \tau^2$ and thus τ^2 is a scaling parameter that controls the variance of the GP.

An example of $l(t; \theta)$ with $\lambda = 1$, $p = 20$, $t_{\max} = 0$ and $\eta = 10$ is plotted in the left panel of Figure 2. Three functions drawn from $\mathrm{GP}(0, K_{\mathrm{Gibbs}})$ are plotted in the middle panel (using the $l(t; \theta)$ as plotted in the left panel) and three drawn from $\mathrm{GP}(0, K_{\mathrm{se}})$ are plotted in the right panel of Figure 2. Clearly, the Gibbs kernel allows more rapid fluctuations around $t = 0$. The functions plotted in the two right panels of Figure 2 can be viewed as draws form a prior GP. As always, with sufficient data the influence of the choice of kernel on the posterior GP is expect to vanish. A further comparison of the GP LC models fitted with the two kernels appears in Section 2.4.3.

2.4 Fitting the Gaussian Processes

The GPs defined by the kernels in (8) and (9) have free parameters that need to be fit from the LC data. A natural way to handle the parameters of the covariance functions is to assign them hyperprior distributions. Letting θ denote the collection of parameters of the kernel, the posterior distribution of θ is

$$p(\theta|y, t) \propto p(y|\theta, t)p(\theta|t), \tag{11}$$

where $p(\theta|t)$ is the prior distribution for θ and $p(y|\theta, t)$ is the marginal likelihood,

$$p(y|t, \theta) = \int p(y, f|t, \theta)\mathrm{d}f = \int p(y|f, t, \theta)p(f|t, \theta)\mathrm{d}f, \tag{12}$$

where the integration marginalizes out the dependence of y on the latent functions, f. We refer to the distribution in (11) as simply 'the posterior', which should not to be confused with the "posterior GP" given in (7).

2.4.1 MAP Estimation

We aim to use the posterior in (11) to estimate the mean function of the LCs in order to classify the SNe. For this purpose a single reasonable estimator of the mean function is sufficient, that is, it is not necessary to carefully quantify uncertainty in the estimate. We use a strategy common in machine learning that uses the maximum a posteriori probability (MAP) estimate, which is the value of θ that maximizes the posterior distribution in (11).

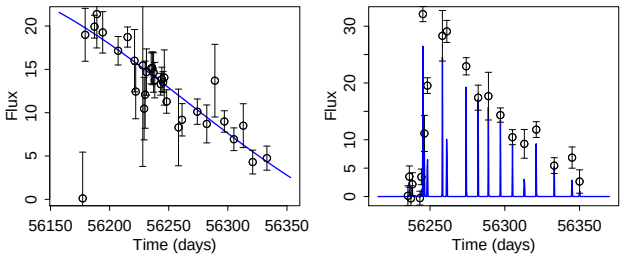

Figure 3. Examples of fitted LCs (i.e. the posterior mean function in (7)) showing pathological MAP estimates under the SE kernel and a flat prior on the kernel's parameters. The fitted LCs are plotted in blue. The left panel is an examples of a very large fitted value of l (r-band of SN 84341 at $z = 0.95$, with $l = 247.2$ and, $\sigma = 18.4$). The right panel is an example of a very small fitted value of l (r-band of SN 92200 at $z = 0.41$, $l = 0.1$, and $\sigma = 15.1$.

To compute the MAP estimate of θ, we first need to derive an expression for the marginal likelihood in (12). This can be accomplished analytically, because the integrand in (12) can be written as the product of two Gaussian densities, specifically, $y|f, t, \theta \sim N(f, \mathrm{diag}(\sigma^2))$ and, given the discrete set of time points, t, we have, $f|t, \theta \sim N\{\mu(t), \mathbf{K}(t, t)\}$. Letting $t = (t_1, \ldots, t_n)^T$, $K = \mathbf{K}(t, t)$, $\mu = \mu(t)$ and $\Sigma = \mathrm{diag}(\sigma^2)$, we thus obtain the marginal likelihood,

$$\begin{aligned}\log p(y|t, \theta) = &-\frac{1}{2}(y - \mu)^T(K + \Sigma)^{-1}(y - \mu) \\ &-\frac{1}{2}\log|K + \Sigma| - \frac{n}{2}\log 2\pi.\end{aligned} \tag{13}$$

Only the first term in (13) involves the data. The second is a penalty for the complexity of the fit. Under the SE kernel, for example, a less complex approximately linear fit has a higher value of l causing the exponential term in (8) to be nearly constant regardless of the distance $(t - s)$. In this case the determinant in the second term of (13) is dominated by the diagonal matrix, Σ, because K is nearly linearly dependent. The final term in (13) is a normalizing constant that depends on the data only through n.

To obtain the log of the posterior in (11) the log-prior can simply be added to (13). Because (13) only depends on θ through K, its derivatives are easily obtained, numerical optimization of (11) is efficient, and it can be easily implemented using gradient-based numerical methods.[2]

2.4.2 Prior Distributions and Fits Under the SE Kernel

Lochner et al. (2016) fitted LCs using a flat prior distribution on the parameters of the covariance function in (11)

[2] We use the constrained gradient based optimizer developed by Byrd et al. (1995) and implemented in `R` through the `optim()` function. This optimizer is based on the quasi-Newton method and only requires the Jacobian (or in our case the gradient), but not the Hessian to be calculated. Instead the Hessian is approximated (Givens & Hoeting 2012, sec. 2.2.2.3).

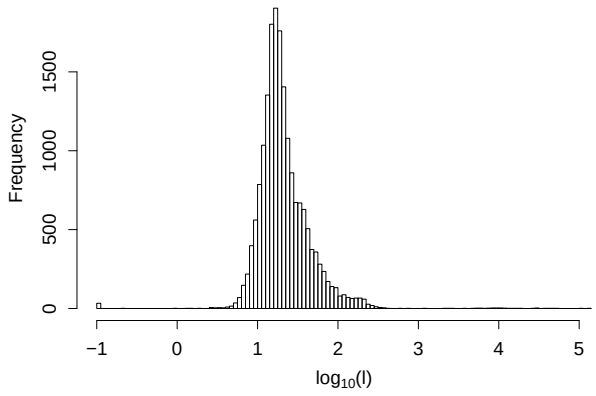

Figure 4. Distribution of the fitted l parameters (MAP estimates) of the r-band fitted with the uniform (improper) prior on l. Notice the extreme values on the log-scale.

(Seikel et al. 2012). This is a common choice in the GP literature (e.g. Rasmussen & Williams (2006), sec. 5.4.1). Although there is in no general guarantee that the posterior is proper with an improper prior, this causes us no difficulty in principle because we only use the MAP estimate and the marginal likelihood is finite.

Unfortunately, using a flat prior gives rise to pathological fits with the data we used. Two examples are plotted in Figure 3. The left plot is an example of a large fitted value for l, causing an almost linear fit. The right plot is an extreme example of problems that occur when the posterior process drifts towards the prior mean, in this case caused by a very small fitted value of l. The histogram in Figure 4 gives the distribution of the fitted (i.e., MAP) values of l in the r-band across the 17,730 SNe (on the log-scale). Extreme values of l in this distribution lead to poor fits like those illustrated in Figure 3. The other bands suffer from similar problems.

The flat prior distribution on τ, on the other hand, does not cause problems. To see this, we rescaled the data by dividing each LC by its largest value, adjusted each σ_i accordingly, and plotted GPs with extreme values in the distribution of the fitted values of τ. No systematic patterns were observed.

To avoid pathological cases like those in Figure 3 requires a proper prior distribution on l. The empirical distribution in Figure 4 obtained using a flat prior inspired the adoption of a log-normal prior distribution,

$$p(l|\nu,\rho) = \frac{1}{\rho\sqrt{2\pi}}\frac{1}{l}\exp\left(-\frac{(\log l-\nu)^2}{2\rho^2}\right). \qquad (14)$$

The distribution has support on the positive real line is right skewed and has, depending on the scale parameter, ν, very low probability for small values. We set the hyper parameters to $\nu = 3.1$ and $\rho = 0.4$, yielding a mean of ≈ 24 and median of ≈ 22, close to the mean and median of the fitted parameters in all bands obtained using a flat prior on l. Using this prior avoid the pathological fits without interfering too much with the GP fits where the flat prior works well. The flat prior on τ is maintained.

The distributions of the fitted values of l across the 17,730 SNe under the log-normal prior distribution are given in Figure 5, for each of the four bands. These histograms are not plotted on the log scale as in Figure 4 and illustrate that this choice of prior avoids the extreme fitted values of l and their associated pathological fitted LCs. The distributions of the MAP estimates of l are quite similar to the prior distribution superimposed on the histograms. This is not a histogram of the posterior distribution of l for a single LC, but the distribution of the l across the LCs. Thus, the similarity between the histograms and the priors does not indicate an overly influential prior distribuiton, but rather that the prior distribution is a relatively good representation of the population distribution of l across SNe (as we would expect, for example, in a hierarchical model). The one exception is the g-band, where the peak in the histogram near the prior mode can be explained by the fact that at higher redshift the g-band maps deeper into the UV, where the SNIa flux drops, leading to fainter emission. This produces more noisy data in this band for many SNe, similar to the SN plotted in Figure 1. With higher noise, the prior dominates the likelihood and hence the posterior distribution peaks near the prior mode.

2.4.3 Prior Distributions and Fits Under Gibbs Kernel

The Gibbs kernel and its length scale function given in (9) – (10) have five free parameters. Initial investigations showed that with the Gibbs kernel the log marginal likelihood in (13) often contains many local maxima. To regularize the fits, we restrict the parameter values, either deterministically or through prior distributions. Some local modes, for example, correspond to the the position parameter, $t_{\max}$, being in regions with noisy data that are away from the peak brightness. This results in over fitting of the noisy regions of the LCs. To avoid this, we restrict $t_{\max}$ to be equal to the value of t that maximizes the fitted LCs under the SE kernel. This restriction does not imply that the Gibbs and SE kernel fits necessarily exhibit a maximum at the same time value, but such values are likely to be similar. We also fix the variance parameter at $\eta = 10$ as in the leftmost panel of Figure 2, restrict p to the interval $[0, 20]$, and restrict λ and τ to be positive. The restriction on p ensures that length scale function, $l(t;\theta)$, is positive.

Since the SE kernel is a special case of the Gibbs kernel (with $p = 0$), the considerations regarding the choice of prior for l discussed in Section 2.4.2 also apply to the scale length, λ, under the Gibbs kernel. Hence we adopt the same log normal prior distribution used for l under the SE kernel for λ. Finally, we use uniform prior distributions for p and τ^2. Because numerical optimizers can easily converge to one of the local maxima, for a number of LCs we obtain a lower value for the log posterior MAP estimate than we did with the SE kernel. In these cases, the SE fits are used instead. (I.e., we set $p = 0$, $\lambda = l$, and $\tau_{\text{Gibbs}} = \tau_{\text{se}}$.) The log posterior of the MAP estimates increases when using the Gibbs kernel (as compared with the SE kernel) for 68%, 75%, 57% and 51% of the LCs in the g, r, i and z bands, respectively.

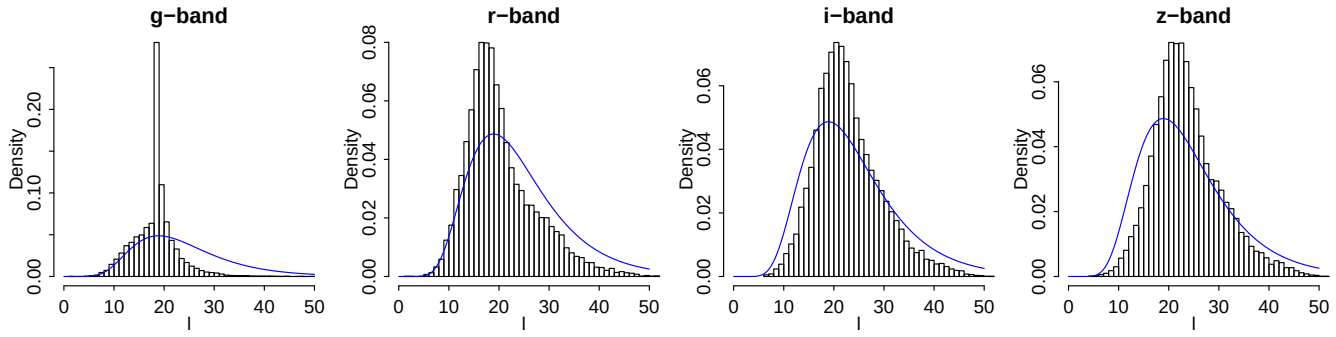

Figure 5. Histograms of fitted l parameters of the SE kernel using the log-normal prior on l, for the entire data set. Superimposed in blue is the log-normal prior distribution.

2.4.4 Comparing the Fits

Figure 6 depicts four representative LCs and their fits under both kernels. The two panels in the first row are good examples of cases that motivate the Gibbs kernel. In both cases, the SE kernel is unable to simultaneously fit the rapid increase followed by a rapid decrease in brightness at the peak and the relatively smooth tails. The result is a relatively small fitted value of l and thus too little smoothing in the tails. The Gibbs kernel in contrast benefits from the flexible length scale function and is able to fit both the peak and the tails well. The LC at the bottom left is an example where both kernels yield very similar fits (although the fitted value of p is not near 0 under the Gibbs kernel). The bottom right LC is an example where the increased flexibility of the Gibbs kernel results in overfitting of relatively noisy data.

Histograms of the difference between the maximum log posterior obtained using the Gibbs kernel and that obtained under the SE kernel appear in Figure 7. (In cases where the log posterior obtained with the Gibbs kernel is less than that obtained with the SE kernel, a zero is recorded in these histograms.) In most cases, the increase in the maximum log posterior value achieved under the Gibbs kernel is small (and thus may be included in the leftmost bins of the histograms in Figure 7.)

Cases of overfitting under the Gibbs kernel, like the one depicted in the bottom right panel of Figure 6, are typically associated with noisy data. In such cases the fits under the two kernels appear quite different while their maximum log posterior values are quite similar. In principle a formal model selection technique (e.g., a Bayes Factor or likelihood ratio test) could be employed to select between the SE and Gibbs kernel but this would require substantial computational effort. Since we do not believe either kernel is "correct" and simply aim for reasonable interpolation of the LCs, we propose a simple method to choose between the kernels. Specifically, to avoid the occasional overfitting associated with the Gibbs kernel, we use this kernel only if it improves the value of the log posterior MAP by at least 2% and using the SE kernal otherwise. In this way we aim to use the Gibbs kernel when it is advantageous (e.g., the cases in the first row of Figure 6), but not when it overfits (e.g, the case in the lower right panel of Figure 6). This strategy results in the Gibbs kernel being used for 17%, 28%, 12% and 8% of the LCs, in the g, r, i and z bands, respectively. In our numerical studies in Section 5 we compare this strategy with (i) using the SE kernel for all SNe and (ii) using the Gibbs kernel for all SNe.

3 NORMALISING THE LIGHT CURVES

The fitted LCs must be aligned in time and normalized in brightness before they are classified. We describe both of these tasks in this section.

3.1 Time Alignment

To align the LCs in time we define time zero to be the time when the fitted LC in the (observer's) i-band achieves its maximum.[3] We choose the i-band because it is the band where most of the fitted LCs (under the SE kernel) have a maximum *within* the time interval of the observations, namely 90.4% of the SNe LCs peaked in the i-band.

3.1.1 Estimating Time Zero for LCs with a Peak

We evaluate the fitted LCs on a one-day spaced grid of time points, starting at the time of the first observation in the i-band. Given that the fitted LCs do not fluctuate much on a daily basis and the computational costs involved with a finer grid, we believe this choice of grid resolution offers a reasonable compromise between computational cost and accuracy. Let $\hat{f}_a(t_j)$ denote the fitted LC of SN a at time t_j in the i-band. (The i-band is suppressed in this notation.) Letting $t_{a,f}$ and $t_{a,l}$ be the times of the first and last observations of SN a in the i-band, respectively, and $\mathcal{T}_a = \{k \in \mathbb{N} \,|\, 0 < k \leqslant t_{a,l} - t_{a,f}\}$, we define time zero as

$$t_{a,0} = t_{a,f} + \underset{k \in \mathcal{T}_a}{\arg\max}\, \hat{f}_a(t_{a,f} + k). \tag{15}$$

[3] This differs from the more standard practice of defining time zero as the time of peak brightness in the rest-frame b-band. Because the b-band is not available in the current data, converting the LCs to the SN rest-frame would require complex K-corrections and accounting for uncertainty in the redshift. Thus, using the observer's frame and the i-band significantly simplifies our procedure.

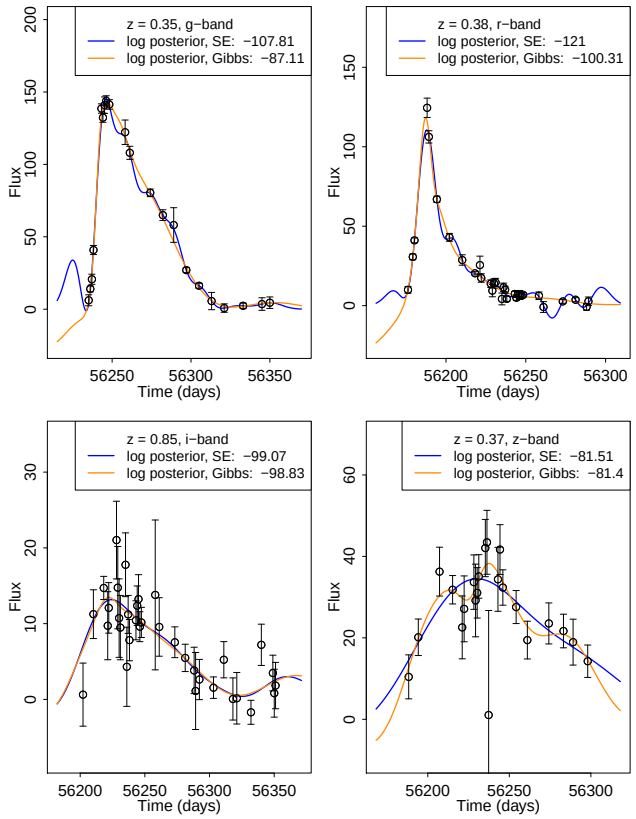

Figure 6. *Top row*: Two examples of LCs (g-band of SN 17125 and r-band of SN 78789, respectively) for which the Gibbs kernel (orange) provides a better fit than does the SE kernel (blue). In both cases the Gibbs kernel is able to fit the peak well without over fitting away from the peak, whereas the SE kernel results in a small length scale and over fits. In both cases the log posterior improved substantially with the Gibbs kernel. *Lower left*: A LC (i-band of SN 113107) where the two fits are quite similar. As expected, the log posterior is similar under the two kernels. *Lower right*: A LC (z-band of SN 147174) where the log posterior of the two fits are almost identical, but the fit under the Gibbs kernel arguably overfits the noisy data. *Fitted Parameters*: Upper left (g-band of SN 17125), SE fit with $l = 7.24$ and $\tau = 58.76$, Gibbs fit with $\lambda = 35.21, \tau = 75.65$, and $p = 20$; Upper right (r-band of SN 78789), SE fit with $l = 7.6$ and $\tau = 34.75$, Gibbs fit with $\lambda = 36.46, \tau = 50.48$, and $p = 20$; Lower left (i-band of SN 113107), SE fit with $l = 23.63$ and $\tau = 6.23$, Gibbs fit with $\lambda = 30.91, \tau = 6.5$, and $p = 9.66$; Lower right (z-band of SN 147174), SE fit with $l = 34.35$ and $\tau = 17.64$, Gibbs fit with $\lambda = 22.11, \tau = 20.62$, and $p = 8.73$.

3.1.2 Estimating Time Zero for LCs without a Peak

The LCs of some of the data were generated with explosion times well before or well after the survey time window, leading to several LCs with missing pre- or post-peak data. Of our 17,330 SNe, after evaluating the fitted LCs on one-day spaced grids (Section 3.1.1) 1,721 do not have a peak in the i-band. For these, time zero is estimated via pairwise comparisons with the peaked LCs. Consider, for example, a pair of SNe, where one has a peak in the i-band and one does not. The LC without a peak is repeatedly shifted according to the one-day spaced grid and its time zero is estimated based on the shift that minimises the mean squared distance between the two LCs. For each LC without a peak, an estimate of this sort is obtained using each of the 15,609 LCs with a peak and the estimates are combined using a weighted average to obtain the final estimate.

We start by considering LCs for which we have data recorded only after the peak. Let a be the index of one such SN and let b be the index of one of the SN with a peak in the i-band. To compare the two LCs, we normalize SN a to its fitted value at the time of its first observation, $\hat{f}_a(t_{a,f})$, and normalize SN b to its fitted value k days past its peak, $\hat{f}_b(t_{b,0} + k)$. (Recall that we aim to estimate the number of days between the unobserved peak for SN a and its first observation.) We then compute the mean square error of the rescaled LCs with a time shift of k days,

$$c_{ab}(k) := \frac{1}{n_a} \sum_{r=0}^{n_a - 1} \left(\frac{\hat{f}_a(t_{a,f} + r)}{\hat{f}_a(t_{a,f})} - \frac{\hat{f}_b(t_{b,0} + k + r)}{\hat{f}_b(t_{b,0} + k)} \right)^2, \tag{16}$$

where $n_a = \lfloor t_{a,l} - t_{a,f} \rfloor$ is the number of grid points for SN a. The estimate of $t_{a,0}$ based on the LC of SN b is the shift (in days) that minimizes the mean square error,

$$\hat{t}^{\text{shift}}_{a,b} = \underset{k \in \{0,1,\ldots,n_{ab}\}}{\arg\min} \; c_{ab}(k), \tag{17}$$

where $n_{ab} = n_b - n_a$ is the difference between the number of grid points. The range of values of k in (17) ensures that $\hat{f}_a$ is evaluated only where it exists in (16). If $n_{ab} < 0$, $\hat{t}^{\text{shift}}_{a,b}$ is set to 0.

Our estimate of time zero for SN a is a weighted average of $\hat{t}^{\text{shift}}_{a,b}$ across all b for which SN b is observed with peak in the i-band; let $\mathcal{P}$ be the collection of indexes, b, of such SNe. In order to favour peaked SNe that (after a shift) align better with SN a, the weights are chosen to be inversely proportional to the mean square error

$$c_{ab} = \left\{ \min_{k \in \{0,1,\ldots,n_{ab}\}} c_{ab}(k) \right\}^{-1}, \tag{18}$$

and our final estimate of time zero is

$$t_{a,0} = t_{a,f} - \frac{1}{\sum_{b \in \mathcal{P}} c_{ab}} \sum_{b \in \mathcal{P}} c_{ab} \, \hat{t}^{\text{shift}}_{a,b}. \tag{19}$$

(Again, if $n_{ab} < 0$, c_{ab} is set to 0.)

A similar method is used to estimate time zero for the LCs observed only before their peaks in the i-band. The only difference is that the LCs in (16) are scaled to have a common right end point, the summation in (16) is from $r = -n_a + 1$ to 0 and the minima in (17) and (18) are taken over $\{-n_{ab}, -n_{ab} + 1, \ldots, 0\}$. This situation happens only for 5.4% of the SNe.

Our method is inspired by and extends that adopted by Richards et al. (2012), who used the shift that minimised the cross correlation between the two LCs rather than (16) and (17), and adopted a simple average rather than a weighted average.

3.1.3 Standardising the Time Domains

We use our estimate for time zero to re-center the time scale for each SN (in all colour bands) at the point of its maximum

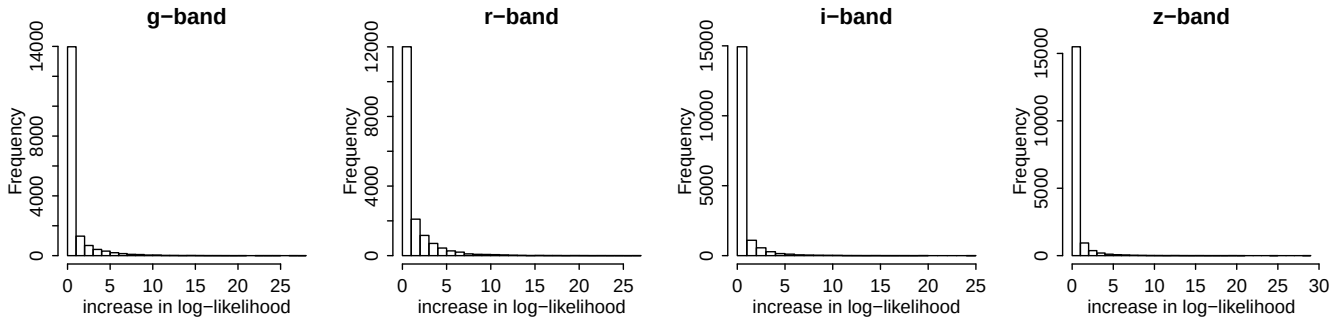

Figure 7. Histograms of the difference between the maximum log posterior obtained using the Gibbs kernel and that obtained using the SE kernel. Values greater than zero indicate an improved fit under the Gibbs kernel. (In cases where the log posterior obtained with the Gibbs kernel is less than that obtained with the SE kernel, a zero is recorded in these histograms.)

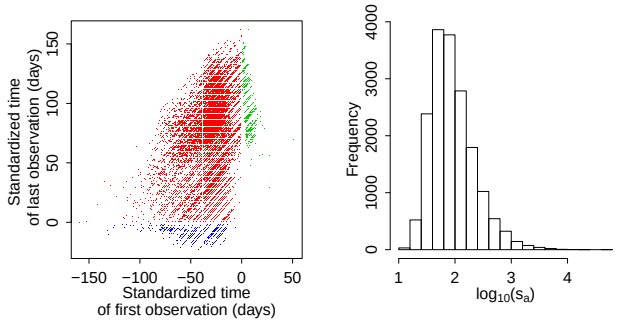

Figure 8. *Left:* Scatterplot illustrating the observed time intervals of the i-band LCs in standardised times, $\tilde{t}$. The plot is colour coded according to whether the i-band peak occurs within the observed LC: blue indicates data is only recorded before the peak, green indicates that data is recorded only after the peak, and red indicates that the peak is observed. *Right:* Histogram of the SNe apparent brightnesses, s_a (arbitrary units), on log-scale (calibrated fluxes are obtained by converting from apparent magnitudes using Eq. (1) in Kessler et al. (2010a)). Both plots are based on the SE kernel.

i-band brightness. Specifically, for each SN, with index a, define a standardized, integer sequence of times in units of days, $(\tilde{t}^c_{a,1}, \ldots, \tilde{t}^c_{a,n^c_a})$, where

$$\tilde{t}^c_{a,1} = \left\lceil t^c_{a,f} - t_{a,0} \right\rceil \quad \text{and} \quad \tilde{t}^c_{a,n^c_a} = \left\lfloor t^c_{a,l} - t_{a,0} \right\rfloor$$

where $n^c_a = \tilde{t}^c_{a,n^c_a} - \tilde{t}^c_{a,1} + 1$ and the superscript, c, indexes the colour bands, $c \in C = \{g, r, i, z\}$.

In left panel of Figure 8 we plot the standardized times of the first and last observation in the i-band to illustrate the time intervals of the observed LCs. The plot is obtained using the SE kernel.

3.2 Standardising the Fluxes

Renormalizing the LCs onto a common brightness scale accentuates certain features that aide classification. (We identify these features using diffusion maps as discussed in Section 4.1.) Following Richards et al. (2012), we divide the LCs of each SN by the sum of the maxima in all of its bands. Specifically, the normalised LC of SN a at time t_j in band c is

$$\tilde{f}^c_a(t_j) = \frac{\hat{f}^c_a(t_j)}{s_a}, \quad \text{with} \quad s_a = \sum_{c \in C = \{g,r,i,z\}} \max_k \hat{f}^c_a(\tilde{t}^c_{a,k} + t_{a,0}), \tag{20}$$

where the maximum is over $k \in \left\{\tilde{t}^c_{a,1}, \ldots, \tilde{t}^c_{a,n^c_a}\right\}$. Thus s_a is the sum of the maxima of the LCs in the four bands. Since the maxima occur at slightly different times for different bands, this quantity does not exactly represent the peak apparent brightness, but acts as a rough proxy for it. A log-scale histogram of the s_a is shown in the right panel of Figure 8 and illustrates the distribution of overall apparent brightnesses among the SNe. (We refer to the s_a as the "brightnesses".) The histogram is based on the fit of the SE kernel.

We denote the set of normalised LCs for SN a by

$$Y_a = \left\{ \left(\tilde{t}^c_{a,j}, \tilde{f}^c_{a,j}\right) | c \in \{g, r, i, z\}, j = 1, \ldots, n^c_a \right\}, \tag{21}$$

where $\tilde{f}^c_{a,j} = \tilde{f}^c_a(\tilde{t}^c_{a,j} + t_{a,0}))$.

4 CLASSIFICATION METHODOLOGY

4.1 Diffusion Maps

The observed SNe are divided into a training and a test set, where the SN types are known in the training set and we aim to identify the types in the test set by comparing SN LCs. Although our GP fits to the LCs allow extrapolation to a common time interval, the degree of variability among the observation intervals (see left panel of Figure 8) means that expanding to a common interval would require substantial extrapolation.

To avoid this, we follow Richards et al. (2012) and apply a diffusion map. This allows us to compute a vector of length m for each LC in such a way that the Euclidean distance between these vectors is a measure of similarity between the LCs. (Formally the distance approximates the so-called diffusion distances between two LCs, as defined in Section A1.) This allows classification of the SNe to be accomplished by applying any "off-the-shelf" classification algorithm to the m dimensional vectors, which thus act as predictor variables for the classification. As discussed in Section 4.2, we use a random forest classifier.

The key advantage of this strategy is that it only requires *pairwise* comparison of the LCs. Thus, the time intervals on which the LCs are compared can be different for each pair; we choose the intersection of the observations intervals for each pair. Although the intersection may be short (or even empty), a further advantage is that the diffusion distance between two LCs is based on the average difference in distances between the two and all other LCs. That is, the diffusion distance between LC a and LC b is a function of the difference in distances $a \leftrightarrow h$ and $b \leftrightarrow h$ averaged over all LCs h. Thus, as long as the observation intervals are not completely separable over the entire dataset, each pair of LCs can be compared in the diffusion space.

In contrast to Richards et al. (2012), we apply the diffusion map to each colour band separately. Thus, for each SN, we obtain four vectors of predictor variables (one for each band). The lengths of these vectors may vary among the bands, and thus we denote the lengths by m_c, for $c \in C = \{g, r, i, z\}$. The advantage of this approach is that the explanatory power of the LCs in the different bands are not blurred with each other. Thus two similar SNe with one or more bands disrupted by noise can still be judged similar by using the unaffected bands. An overview of diffusion maps and the details of our choice of the diffusion distance are given in Appendix A.

4.2 Random Forest Classification

Random forest is a classification algorithm that is based on "growing" replicate classification trees using bootstrapped samples of the data. Each classification tree is grown by recursively splitting the data into binary regions based on one of the predictor variables. (Recall the predictor variables are generated by the diffusion map in our case.) In each split, the variable and split point is chosen such that the sum of the *gini* indexes[4] in the two new regions is decreased as much as possible. In each split only a randomly selected subset of m_{try} of the predictor variables are considered. The trees are grown to their maximum size, which means that the recursive partitioning of the predictor variables into subregions is stopped only when one class (of SNe) is present in each region, e.g. because there is only one SN left in the region. Once all of the trees are grown, the classification of SN from the test set is based on 'voting' of the trees. Since we have only two classes, SNIa and not SNIa, we can use a simple threshold, γ_{Ia}, where we classify a SN as SNIa if the proportion of trees so voting is greater than γ_{Ia}.

Random forests are relatively robust to noisy predictor variables as long as there is a reasonable number of relevant variables (Friedman et al. 2009). Therefore, all the predictor variables from the four optimised diffusion maps (i.e., up to 100) are combined into a vector and passed to the random forest without further selection.

[4] With two classes, the gini index of a region is defined as $2p_{\text{Ia}}(1 - p_{\text{Ia}})$, where p_{Ia} is fraction of SN that are SNIa in the region. Thus, the gini index will be minimised if nearly all or nearly none of the SNe in the region are SNIa.

4.3 Tuning the Diffusion Maps and Random Forest Classifier

To implement the diffusion map and random forest, we must set their tuning parameters, $(\varepsilon_g, \varepsilon_r, \varepsilon_i, \varepsilon_z)$, m_{try}, and γ_{Ia}. (See (A1) in Appendix A for a definition of ε_c; there is a separate value of ε_c for each colour band, $c \in C$.) We aim to set these parameters in such a way as to optimise classification, while recognizing that classifying a SN as a SNIa when it is not (i.e. a false positive) is worse than neglecting to identify a SN as a SNIa (i.e., a false negative), at least in the typical application of using SNIa as standard candles. For this reason, Kessler et al. (2010a) proposed the following criterion:

$$\zeta_{\text{Ia}}(W) = \frac{N_{\text{Ia}}^{\text{True}}}{N_{\text{Ia}}^{\text{Total}}} \times \frac{N_{\text{Ia}}^{\text{True}}}{N_{\text{Ia}}^{\text{True}} + W N_{\text{Ia}}^{\text{False}}}, \tag{22}$$

where $N_{\text{Ia}}^{\text{Total}}$ is the total number of SNIa, $N_{\text{Ia}}^{\text{True}}$ is the number of correctly classified SNIa (true positives), and $N_{\text{Ia}}^{\text{False}}$ is the number of SNe incorrectly classified as a SNIa (false positives). The first term on the right hand side of (22) is also known as the efficiency, e_{Ia}, of the classifier, i.e., the proportion of the SNIa that are correctly identified. For $W = 1$, the second term of (22) is known as the purity, p_{Ia}, i.e., the proportion of true positives among the overall number of positives. In the classification challenge W was set to 3 (Kessler et al. 2010a, Section 5) meaning that false positives incur a heavier penalty than false negatives. For simplicity, we fix $\zeta_{\text{Ia}} = \zeta_{\text{Ia}}(3)$.

We set the tuning parameters in two steps. First we set each of the $(\varepsilon_g, \varepsilon_r, \varepsilon_i, \varepsilon_z)$ separately, using only its corresponding diffusion map and predictor variables. Starting with one of the bands, call it band c, we optimise ζ_{Ia} over a grid of values of ε_c. Then, for each value of ε_c in the grid, ζ_{Ia} is further optimized over a grid of proportions, γ_{Ia}. These optimisations are conducted using random forests of 500 trees and with the default value of $m_{\text{try}} = \lfloor \sqrt{m_c} \rfloor$, where m_c is the number of predictor variables corresponding to the current band. To reduce over fitting, only the out of bag (OOB) predictions from the training set are used when computing (22). The OOB prediction of each SN (in the training set) is based only on the trees that were fit *without* that SN, i.e., trees for which that SN was not included in the bootstrap sample. In this way the OOB prediction is similar to a "leave one out" cross validation procedure.

In the second step, we fix $(\varepsilon_g, \varepsilon_r, \varepsilon_i, \varepsilon_z)$ at the values derived above and compute the final optimal values of γ_{Ia} and m_{try}. In this step we use the combined predictor variables from all four bands. Again we optimise ζ_{Ia} computed using OOB predictions from the training set, this time over a grid of values of m_{try} and γ_{Ia}.

5 CLASSIFICATION RESULTS

We now describe the classification results obtained using the two GP kernels described in Section 2.3. We compare three GP kernel fits, given in Table 2: the first exclusively uses the SE kernel; the second exclusively uses the Gibbs kernel; and the third uses the Gibbs kernel whenever it improves the log posterior of the MAP estimate by more than 2% over the SE kernel and uses the SE kernel otherwise. For each choice we train separately on the two training sets described in Section

2.1, namely the biased training set, $\mathcal{B}_{\rm train}$, and the unbiased training set, $\mathcal{U}_{\rm train}$. In each case the results are evaluated on the appropriate test sets. We call the combined choice of GP kernel and training set a "model".

5.1 Tuning the Diffusion Map and Classifier

Proceeding as described in Section 4.3, the optimal parameters for the diffusion maps and random forest classifier under each model appear in Table 3. In the table, we also show the relative weight of each band, quantified by

$$\hat{I}_c = \frac{\sum_{v\in\mathcal{V}_c}\hat{I}_v}{\sum_{v\in\mathcal{V}}\hat{I}_v}, \quad \text{for} \quad c\in\{g,r,i,z\}, \tag{23}$$

where $\mathcal{V}$ is the complete set of predictor variables generated by all four diffusion maps, $\mathcal{V}_c$ is the subset of $\mathcal{V}$ corresponding to the variables generated by the diffusion map for colour band c, and $\hat{I}_v$ is the "importance" associated with variable v. The importance of each variable is determined as the trees of the random forest grow. Specifically, the decrease in gini index associated with each split is recorded alongside the variable that was divided into two regions in each partition step and the variable importance, $\hat{I}_v$, is computed by separately summing the decreases in gini index associated with each variable (Friedman et al. 2009, p 593).

The diffusion coordinates of the two most important variables as judged by the random forests are plotted and colour coded according to SN type for both the training and test sets under models $\mathcal{B}_1$ and $\mathcal{U}_1$ in Figure 9. (The other models are qualitatively similar.) While we plot only the two most important diffusion coordinates, we use up to 25 coordinates (in each band) in the classification, see Appendix A for details. The separation between the three SN types is relatively good in $\mathcal{B}_{\rm train}$ but less so in $\mathcal{B}_{\rm test}$. Although $\mathcal{U}_{\rm train}$ shows a higher degree of superposition between different types than does $\mathcal{B}_{\rm train}$, the separation that does exist for $\mathcal{U}_{\rm train}$ does not degrade in $\mathcal{U}_{\rm test}$. We emphaise that the random forests are only trained to distinguish SNIa from non-SNIa and hence do not distinguish between sub-categories of non-SNIa (i.e., the green and blue dots in Figure 9).

5.2 Classification Results

Classification results appear in Table 5. The first three columns summarise results for the training sets, where the optimality criterion, $\zeta_{\rm Ia}$, the purity, $p_{\rm Ia}$, and the efficiency, $e_{\rm Ia}$, are estimated from the OOB samples. The last four columns summarize results for the corresponding test sets. The last column reports the area under the ROC (Receiver Operating Characteristic) curve, which is defined and discussed below. Where appropriate, our results are compared with previous results in the literature in Table 4.

Models $\mathcal{B}_1$–$\mathcal{B}_3$ obtain high purity and efficiency values for $\mathcal{B}_{\rm train}$, with all of the purity values and two of the three efficiency values being above 95%. The small differences among the models are likely due to the values of the threshold $\gamma_{\rm Ia}$. In particular, Models $\mathcal{B}_2$ and $\mathcal{B}_3$ have higher values of $\gamma_{\rm Ia}$ and higher purity, but lower efficiency. Unfortunately, for all three models performance degrades substantially on $\mathcal{B}_{\rm test}$, with purities ranging from 0.57 for model $\mathcal{B}_1$ to 0.65 for model $\mathcal{B}_3$ and efficiencies between of 0.80 and 0.88. Because poor purity is penalised harshly, $\zeta_{\rm Ia}$ drops from > 0.86 in $\mathcal{B}_{\rm train}$ to < 0.31 in $\mathcal{B}_{\rm test}$.

All of the models using $\mathcal{U}_{\rm train}$ preform worse than their counterparts which use $\mathcal{B}_{\rm train}$ in terms of both purity and efficiency on their respective training sets. This is not unexpected given the smaller separation of the SNIa and non-SNIa in $\mathcal{U}_{\rm train}$ relative to $\mathcal{B}_{\rm train}$. (Compare the first and third panels of Figure 9.) Unlike Models $\mathcal{B}_1$–$\mathcal{B}_3$, however, performance does not degrade substantially when Models $\mathcal{U}_1$–$\mathcal{U}_3$ are applied to their test set. If a classifier is tuned too precisely to the training set, its performance tends to degrade substantially in the test set. Using OOB samples aims to avoid this and appears to work quite well, at least with an unbiased training set.

The degradation of the models $\mathcal{B}_1$–$\mathcal{B}_3$ when applied to $\mathcal{B}_{\rm test}$ illustrates a challenge that arises when the test set is not representative of the larger population. (This has been noted by others in this context, see Richards et al. (2012); Varughese et al. (2015); Lochner et al. (2016)) The good performance on the test set when the classifiers are trained on $\mathcal{U}_{\rm train}$, however, highlights the potential of our classification scheme. This gives us a guiding principle for developing an improved method: the bias in the training set must be addressed; this is the topic of Section 6.

Table 5 compares the performance of each model using a single classification threshold, $\gamma_{\rm Ia}$, for each. To better explore the trade-off between purity and efficiency, we can plot efficiency against the false positive rate (the number of false positives relative to the total number of negatives, i.e. non-Ia) of the predictions using a fine grid of $\gamma_{\rm Ia}$ values between 0 and 1. Such plots are known as Receiver Operator Characteristic (ROC) curves and appear in Figure 10. A perfect classifier would have an efficiency of one and a false positive rate of zero and would appear in the upper left corner of the panels in Figure 10. The ROC curve of a random classifier, on the other hand, would appear as a 45 degree line, as shown as a dotted black line in Figure 10. The area under the ROC curve (AUC) is a simple summary of overall performance. A perfect classifier would have an AUC of 1, while a random classifier would have an AUC of 0.5.

As expected, the ROC curves for $\mathcal{U}_{\rm test}$ are above those for $\mathcal{B}_{\rm test}$, indicating better performance. Likewise the AUC values in Table 5 indicate better performance with $\mathcal{U}_{\rm test}$. Comparing both the ROC curves and the AUC values for the three different GP kernels, however, shows no substantial difference between the GP models. Thus, the differences in the kernel LC fits (e.g., in Figure 6) somewhat surprisingly do not translate to differences in classification performance.

Figure 11 plots the performance of the models as a function of redshift (left panels) and as a function of the log brightnesses, $\log(s_a)$ (right panels). (Recall that $\log(s_a)$ is a measure of apparent brightness for SN a, see (20).) The classifiers trained on $\mathcal{B}_{\rm train}$ perform equally well on $\mathcal{B}_{\rm test}$ at all redshifts up to $z \lesssim 0.9$, but efficiency and purity drop for larger z (top left panel). For classifiers trained in $\mathcal{U}_{\rm train}$, performance on $\mathcal{U}_{\rm test}$ increases to $z \approx 0.9$, mainly due to increasing efficiency, and then performance drops at the highest redshift values (bottom left panel). The efficiency is generally lower than purity with $\mathcal{U}$, and vice-versa with $\mathcal{B}$. This can be attributed to the different classification thresholds, $\gamma_{\rm Ia}$, which in turn is due to the biased composition of $\mathcal{B}_{\rm train}$.

As a function of brightness the classifiers trained on

Designation	Kernel	Note
1	SE	Applied to all LCs
2	Gibbs	Applied to all LCs
3	Gibbs / SE	Gibbs where log posterior increased by more than 2%

Table 2. GP kernels compared in Sections 5 and 6.

Model	$(\varepsilon_g, \varepsilon_r, \varepsilon_i, \varepsilon_z)$	m_{try}	γ_{Ia}	$\hat{I}_g$	$\hat{I}_r$	$\hat{I}_i$	$\hat{I}_z$
$\mathcal{B}_1$	$(10, 2, 10, 20) \times 10^{-6}$	10	0.57	0.32	0.38	0.14	0.16
$\mathcal{B}_2$	$(10, 2, 10, 10) \times 10^{-6}$	9	0.67	0.28	0.40	0.17	0.15
$\mathcal{B}_3$	$(10, 2, 20, 20) \times 10^{-6}$	10	0.61	0.27	0.42	0.16	0.14
$\mathcal{U}_1$	$(2, 2, 2, 6) \times 10^{-4}$	9	0.43	0.20	0.30	0.27	0.23
$\mathcal{U}_2$	$(5, 5, 2, 8) \times 10^{-4}$	9	0.44	0.20	0.26	0.29	0.25
$\mathcal{U}_3$	$(3, 2, 2, 2) \times 10^{-4}$	12	0.45	0.19	0.27	0.27	0.28

Table 3. Optimized tuning parameters for the diffusion maps and random forests (i.e., ε_c and m_{try}), classification threshold, γ_{Ia}, and relative importance of the different bands, ($\hat{I}_c$, defined in Eq. (23)), in the final combined random forests, for $c \in \{g, r, i, z\}$. The model subscripts refer to the designations in Table 2.

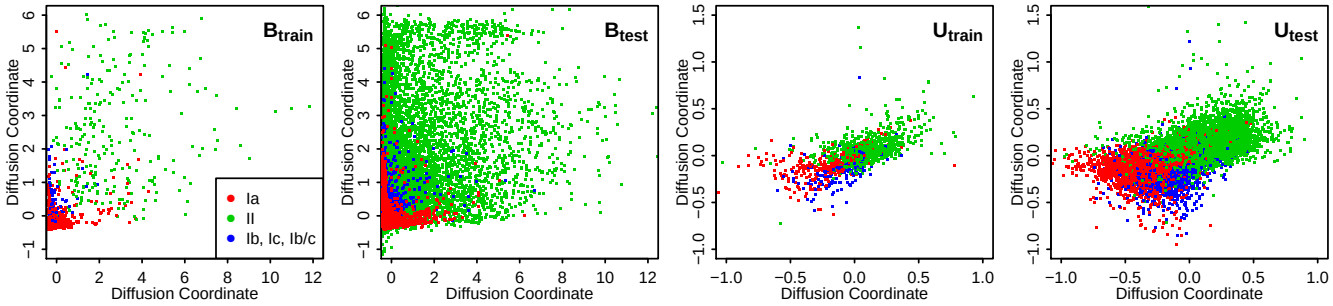

Figure 9. The two most important diffusion coordinates for models $\mathcal{B}_1$ and $\mathcal{U}_1$. *Left Two Panels*: Diffusion coordinates for the test and training sets used with $\mathcal{B}_1$. *Right Two Panels*: Diffusion coordinates for the test and training sets used with $\mathcal{U}_1$. For the two test sets, the diffusion coordinates are computed after applying the Nyström extension of Eq. (A8). The separation between SNIa and non-SNIa in $\mathcal{U}_{\text{train}}$ is much better maintained in $\mathcal{U}_{\text{test}}$ than the separation in $\mathcal{B}_{\text{train}}$ is maintained in $\mathcal{B}_{\text{test}}$.

		ζ_{Ia}		AUC	
	Dataset	Biased	Unbiased	Biased	Unbiased
STACCATO	Original	0.53	0.59	0.96	0.98
Kessler et al. (2010b)†	Original	$0.3 - 0.45$	–	–	–
Newling et al. (2011)	Original	0.39	–	–	–
Newling et al. (2011)	Updated	0.15	0.45	–	–
Richards et al. (2012)	Updated	0.13	0.31⋆	–	–
Varughese et al. (2015)	Original	0.49	0.55	–	–
Lochner et al. (2016)	Updated	–	–	~ 0.88	0.98

†Summary of the performance achieved by the "most stable" (as a function of redshift) classifiers entered in the original challenge.
⋆Richards et al. (2012) used a resampling strategy that does not produce a truly unbiased training set.

Table 4. Best results for various SNe classification methods. Although none of the entries are strictly comparable and comparisons should only be taken as a rough guide, STACCATO results are superior to those obtained with previously published methods, as evidenced by its higher optimality criteria and AUC values.

Model	$\gamma_{\rm Ia}$	Training set $\tilde{\zeta}_{\rm Ia}$	$\tilde{p}_{\rm Ia}$	$\tilde{e}_{\rm Ia}$	Test set $\zeta^*_{\rm Ia}$	$p^*_{\rm Ia}$	$e^*_{\rm Ia}$	AUC*
$\mathcal{B}_1$	0.57	0.88	0.97	0.97	0.27	0.57	0.88	0.93
$\mathcal{B}_2$	0.67	**0.88**	0.98	0.93	0.31	0.65	0.80	0.93
$\mathcal{B}_3$	0.61	0.87	0.97	0.95	0.28	0.59	0.85	0.93
$\mathcal{U}_1$	0.43	0.60	0.88	0.86	**0.59**	0.87	0.86	0.98
$\mathcal{U}_2$	0.44	0.56	0.88	0.80	0.59	0.88	0.83	0.98
$\mathcal{U}_3$	0.45	0.58	0.88	0.84	0.58	0.87	0.84	0.98

˜denotes OOB estimates from the training set.
* based on predictions of the test set.

Table 5. Results of classifying SNIa using different models. Here $\zeta_{\rm Ia}$ is the training criterion (22) with $W = 3$, $p_{\rm Ia}$ is the purity, $e_{\rm Ia}$ the efficiency, and AUC is the area under the ROC curve. The best results on $\zeta_{\rm Ia}$ in the training sets and test sets are highlighted in bold. The classification threshold, $\gamma_{\rm Ia}$, from Table 3 is also shown, to illustrate its influence on the three classification measures.

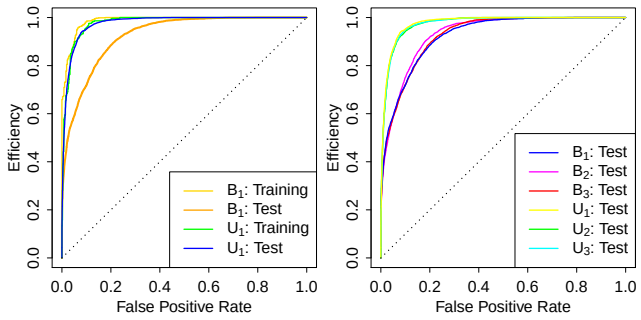

Figure 10. *Left*: ROC curves for models $\mathcal{B}_1$ and $\mathcal{U}_1$ on both training and test sets. *Right*: ROC curves for all models on test data only.

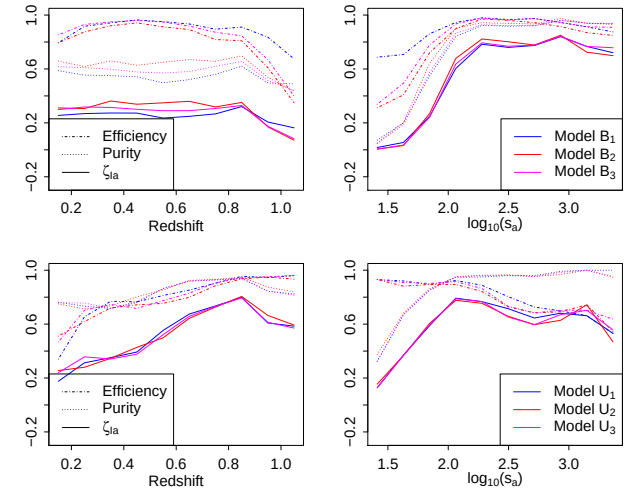

Figure 11. *Top row*: Performance of models trained on $\mathcal{B}_{\rm train}$ when used to classify $\mathcal{B}_{\rm test}$, as a function of redshift and as a of function of s_a, the log brightnesses. *Bottom row*: Same but for models trained on $\mathcal{U}_{\rm train}$ and tested on $\mathcal{U}_{\rm test}$.

$\mathcal{B}_{\rm train}$ perform well on the brighter SNe (larger values of $\log s_a$). For SNe with $\log s_a > 5$, both the purity and efficiency for the best model are above 90%. However, with fainter SNe the performance is much poorer with very low purity levels. For the models trained on $\mathcal{U}_{\rm train}$ the same qualitative description holds, although efficiency is higher for fainter SNe.

Finally, we check whether the number of observation times in the LCs influences the performance of the classifiers. Surprisingly classification is equally good with SNe with many observations and with those with only few.

The overall picture emerging from this analysis is that the classifiers struggle with dimmer, higher-redshift SNe, more so when trained with $\mathcal{B}_{\rm train}$. To improve performance, we must account for the bias in the training set and improve training for high redshift/low brightness SNe. This is the guiding idea behind our STACCATO method, which we introduce in the next section.

6 IMPROVING THE CLASSIFIER WITH STACCATO

The SN classification challenge of Kessler et al. (2010a) provided a biased training set because this is what is expected in a real sample of spectroscopically confirmed SNe. Dealing with the issue of selection bias is a common problem in astronomy, and it was recognized in the early 20th century. Today, astronomers often refer to Malmquist (1925) bias to indicate truncation in magnitude-limited samples; and to Eddington (1913) bias to indicate the effect of measurement error near the sensitivity limit of a survey, when the underlying sources have a steep distribution of the latent property one is trying to measure. The usual way to address selection bias is by "correcting" for it via simulation (in cosmological SN analysis, see e.g. Kessler & Scolnic (2017)), although arguably better Bayesian solutions exist (e.g., Budavari (2009); Rubin et al. (2015); Kelly (2007); Gelman et al. (2013)). Here, we are interested in the specific issue of how to overcome the effect of selection bias in the training sample, which thus become non-representative of the full (test) sample. In other contexts, data augmentation schemes have been proposed to address this problem (e.g., Hoyle et al. (2015)). In this section, we present a new method that aims to improve the performance of classifiers that use only a biased training set, $\mathcal{B}_{\rm train}$. In particular, in Section 6.1 we use propensity scores to partition $\mathcal{B}_{\rm train}$ and then suggest a novel approach in Section 6.2 that uses the partition to improve classification. We call our approach "Synthetically Augmented Light Curve Classification" or STACCATO. Given the similarities between classifiers based on the GP kernels considered in Section 5.2, STACCATO is implemented only with the SE kernel and only with $\mathcal{B}_{\rm train}$. It could, however, be applied to any set of GP fitted LCs using any similarly biased training set.

6.1 Grouping SNe by Propensity Scores

Figure 12 illustrates the excess of bright SNe in $\mathcal{B}_{\rm train}$, especially at high redshift. The left panel shows a scatter plot of redshift and brightness (in terms of $\log s_a$, from Eq. 20) for a random sample of $\mathcal{B}_{\rm test}$; the right panel plots the same

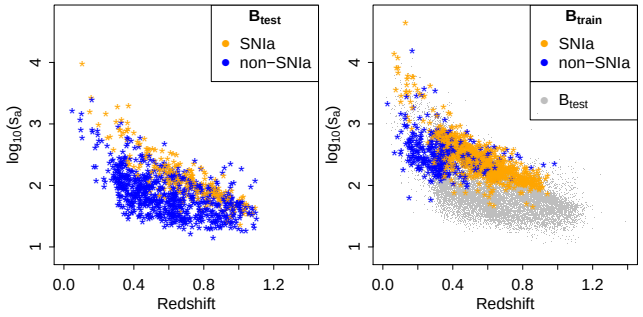

Figure 12. *Left*: Scatterplot of $\log s_a$ against redshift for a random subset of 1,000 SNe from $\mathcal{B}_{\text{test}}$. *Right*: Same plot for SNe in $\mathcal{B}_{\text{train}}$. Both panels are colour coded by SN type. In the right panel the grey dots represent the test data for better comparison of the domains.

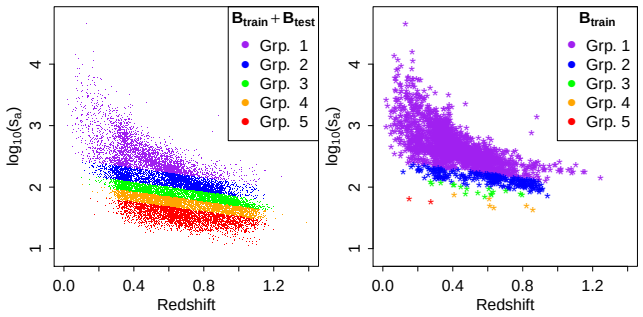

Figure 13. Propensity Score Groups. *Left*: Scatterplot of $\log_{10} s_a$ against redshift for the entire dataset ($\mathcal{B}_{\text{train}}+\ \mathcal{B}_{\text{test}}$), colour-coded according to the five groups in Table 7, obtained by partitioning on the fitted propensity score. *Right*: The same, but showing only the data from $\mathcal{B}_{\text{train}}$. One can clearly see the paucity of training data in groups 3–5.

variables for $\mathcal{B}_{\text{train}}$. Clearly, $\mathcal{B}_{\text{train}}$ over-represents brighter SNe and completely lacks the fainter SNe found in $\mathcal{B}_{\text{test}}$, especially at high redshift.

Propensity scores are used in observational studies to quantify the probability that a unit is in the treatment group rather than the control group (Rosenbaum & Rubin 1984). Here we define the propensity score as the probability that a particular SN belongs to $\mathcal{B}_{\text{train}}$, as function of a set of observed covariates. We can partition the data based on their propensity scores into a number of groups, such that within these groups the bias between the training and test set is substantially reduced. Rosenbaum & Rubin (1984) argue that (under certain assumptions) five groups are sufficient to remove approximately 90% of the bias. Propensity scores can naturally be modelled using a logistic regression with the response variable being membership in $\mathcal{B}_{\text{train}}$. Ideally all covariates that exhibit bias (i.e., a different distribution in $\mathcal{B}_{\text{train}}$ and $\mathcal{B}_{\text{test}}$) should be included in the regression so as to balance their distributions in the groups. For illustration we include redshift and brightness in the logistic regression and define the propensity score to be the conditional probability of a randomly selected SN being a member of $\mathcal{B}_{\text{train}}$ given its brightness and redshift. Table 6 reports the fitted (maximum likelihood) coefficients for the logistic regression.

Predictor variable	Estimate	p-value
(Intercept)	-10.6790 ± 0.3348	$< 10^{-4}$
Redshift	1.3412 ± 0.2009	$< 10^{-4}$
Brightness	1.4751 ± 0.0466	$< 10^{-4}$

Table 6. Logistic regression applied to the entire dataset (n=17,330). The response variable identifies which SNe are in $\mathcal{B}_{\text{train}}$ (= 1) and which are in $\mathcal{B}_{\text{test}}$ (= 0). (Apparent) Brightness is described by $\log(s_a)$ from Eq. (20). We also show the p-value for including each predictor variable in the model, given the others.

Group	Set	Number of SNe	Number of SNIa	Proportion of SNIa
1	Training	947	652	0.69
	Test	2519	1242	0.49
2	Training	245	181	0.74
	Test	3221	1147	0.36
3	Training	17	12	0.71
	Test	3449	754	0.22
4	Training	6	6	1
	Test	3460	342	0.10
5	Training	2	0	0
	Test	3464	107	0.03

Table 7. Composition of the five groups obtained by partitioning the data based on the fitted propensity scores.

Following Rosenbaum & Rubin (1984), we use the quintiles of the estimated propensity scores (i.e., the fitted values from the logistic regression) to split the entire data set into five equally-sized groups. The resulting partition into groups is plotted in Figure 13 and summarised numerically in Table 7. Due to the severe bias in brightness, Groups 3–5 contain very few SNe from the training set, $\mathcal{B}_{\text{train}}$: specifically, they contain just 17, 6, and 2 of the 1,217 SNe in $\mathcal{B}_{\text{train}}$. Notice that even though the SNe types are not used in the linear regression, the type imbalances between the test and training sets are greatly reduced in Groups 1–2 compared to the imbalance in the test and training sets as a whole (cf. Table 1).

In Figure 14 ROC curves for models $\mathcal{B}_1$ (left) and $\mathcal{U}_1$ (right) are plotted for the five propensity score groups from Table 7, where the grouping is only used for partitioning the test set; the random forest classifier is fit and tuned using the combined data. Under model $\mathcal{B}_1$, the ROC curves become coarser with increasing group number because of the smaller numbers of SNIa in the Groups 3–5. The performance in Groups 1–2, where a large proportion of $\mathcal{B}_{\text{train}}$ resides, is very good with almost perfect classification. Performance degrades in Groups 3–5, however, to the degree that classification in Group 5 is not much better than a random classifier.

When using an unbiased training set (right panel of Figure 14) classification is nearly equally good in all five groups, with only slightly worse classification in Group 5. This recalls previous results that conclude that an unbiased training

Term	Meaning
Groups	The entire dataset is split into five equal sized groups based on the propensity scores.
Test (Training) Groups	SNe in the test (training) set are partitioned according to their group membership.
Augmented Training Groups	The training groups are augmented with other training groups and/or synthetic LCs before being used to train the group-specific classifiers.
Validation and Generalization Groups	Each test group is randomly divided into a validation and generalization group, such that each validation groups contains 1500 SNe.

Table 8. Terminology used for group-specific analyses.

set is necessary to obtain good performance with standard classification methods (Lochner et al. 2016; Varughese et al. 2015). In the next section we show how propensity scores can be used to obtain good classification even with a biased training set.

6.2 STACCATO: Synthetically Augmented Light Curve Classification

6.2.1 STACCATO methods

Our approach to mitigate bias and improve LC classification is to train the classifier separately within the five groups defined by the propensity scores. Thus we divide both the test and training set into five groups according to the propensity scores to obtain the five *test groups* and five *training groups*, respectively (See Table 8 for group terminology.) An immediate difficulty with obtaining group-specific classifiers is the lack of data in Training Groups 3–5, and in particular the lack of SNIa. In situations with imbalanced data in one or both categories, a popular strategy is to oversample the underrepresented observations and/or to down sample the over-represented observations (Japkowicz et al. 2000). Rather than resampling the sparse training groups (corresponding to Groups 3–5), we create *augmented training groups* by synthesizing observations in the LC space, a procedure inspired by Ha & Bunke (1997); Chawla et al. (2002) – an approach we call 'STACCATO'.

We thus exploit the probabilistic nature of the GPs to over-sample the LCs by sampling 'synthetic' curves from the fitted GPs. More precisely, we augment the small training groups by generating K synthetic LCs for each SN in that group, where the *augmentation factor*, K, may vary between training groups and where each synthetic LC is drawn band-by-band from the fitted GP for that SN. Compared with simply resampling Training Groups 3–5, our augmented training groups have two advantages. First, the posterior mean vectors of the GPs that we use as the input for the diffusion maps are only estimates and may differ appreciably from the true LCs, at least with noisy data. By sampling LCs from the GPs, the augmented training group takes these uncertainties into account. Second, sampling additional LCs creates a richer sample in the regions of diffusion space most relevant for classification.

The key innovation of STACCATO is to construct an augmented training group for each of the five test groups. The augmented training group for each test group may be constructed using more than one of the original training groups, and each may be supplemented with synthetic LCs using different augmentation factors. As there are many ways to augment the training groups, choosing the augmentation scheme may benefit from an optimization step. The rows of Table 9 give the augmented training groups configurations considered for each of the five test groups. Here, a '+' indicates that the SNe of a given training group (columns) are included in the augmented training set for that test group (rows); a '–' indicates that they are not included; a '+/–' indicates that both possibilities are considered; and the numbers in parentheses are the considered augmentation factors, i.e., the number of synthetic LCs that are sampled for each SN of the original training group. While this is not an exhaustive search of the possible configurations, it is meant to span a wide range of choices. We then select the optimal configuration as the one that gives the highest AUC within each test group, as described below.

The sampled synthetic LCs are standardized in exactly the same manner as the original data. For each possible configuration of the augmented training groups, the diffusion maps are constructed using only the LCs in that configuration. For simplicity we set $\epsilon = 2 \times 10^{-5}$ in all diffusion maps and $m_{\text{try}} = \lfloor \sqrt{\sum_{c \in C} m_c} \rfloor$ in all random forests, where $C = \{g, r, i, z\}$. (I.e., the diffusion maps and random forests are *not* optimized over ϵ and m_{try} as described in Section 4.3.) For each of the five groups, we want to select the configuration that gives the highest AUC within that group. In order to accomplish this, we follow Friedman et al. (2009, ch. 7) and partition the test group into a validation and a generalisation group, with 1500 LCs in each validation group. The validation groups are used to compute the AUC for each configuration, and the configuration that leads to the highest AUC is chosen. The generalization groups are used to measure the performance of the best configuration. In practice, the SN types in the test groups are not known so the training groups or subsets of the training groups should instead be used to compare the AUC for each configuration. This, however, may be infeasible in practice when the training groups are very small (e.g., Training Groups 3–5). Alternatively, the optimal configuration choice can be determined using a simulation study, such as that used in the SN photometric classification challenge.

6.2.2 STACCATO summary

In summary, STACCATO proceeds by

(i) Fitting the LCs using a GP implemented with a squared exponential kernel as described in Section 2.2;

(ii) Forming the training groups by grouping the training set according to a fitted propensity score model as described in Section 6.1;

(iii) Augmenting each of the training groups with observed LCs from other training groups and/or synthetic LCs sampled under the GP fits of observed SNe, as described in Section 6.2.1;

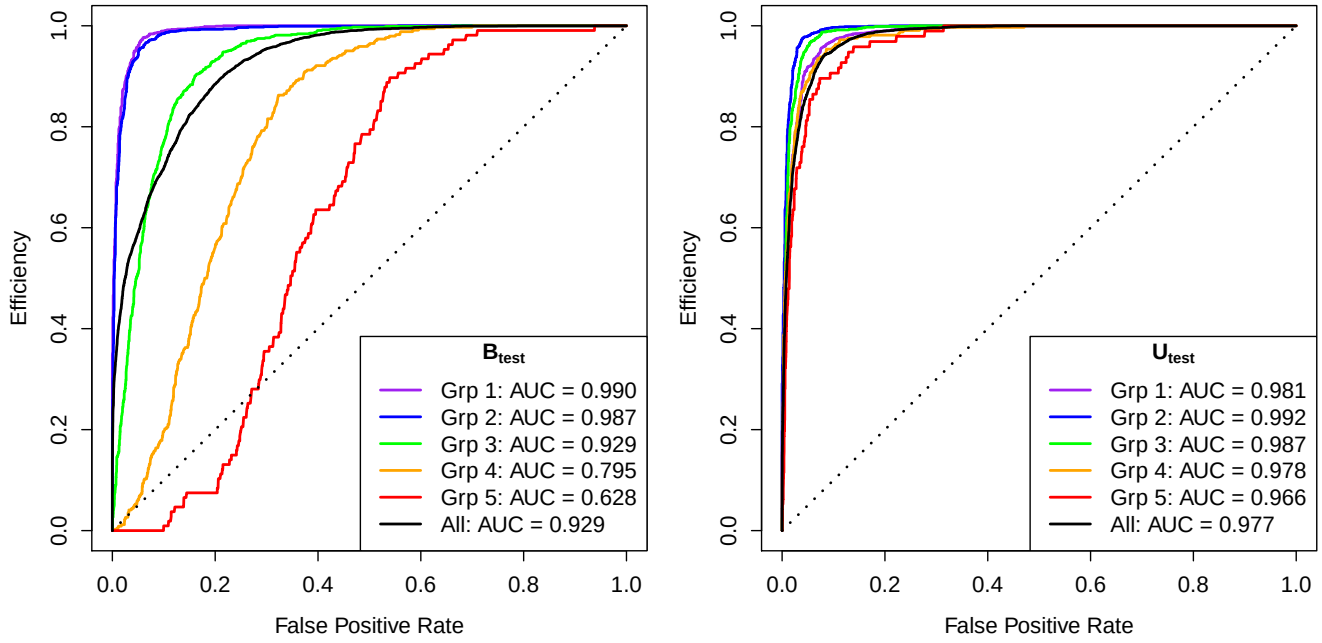

Figure 14. ROC curves for the five test groups in Table 7, computed using a classifier trained with the biased training set (right) and with the unbiased training set (left). In both cases the grouping is only used for partitioning the test set. The random forest classifier is fit and tuned using the combined data.

Test Group	Training Groups 1		2		3		4		5	
1	+	(0)	+/−	(0-2)	−		−		−	
2	+/−	(0)	+	(0-2)	−		−		−	
3	−		+/−	(0)	+	(0-10)	+	(0-10)	+	(0-10)
4	−		+/−	(0)	+	(0-10)	+	(0-10)	+	(0-10)
5	−		+/−	(0)	+	(0-10)	+	(0-10)	+	(0-10)

Table 9. Composition of the augmented training groups (along rows) for each test group (rows). A '+' indicates that an augmented training group included the original training group corresponding to a particular column; a '−' indicates that original training group is not used; a '+/−' indicates that both combinations were tried. The numbers in parentheses give the number of LCs synthesized for each of the LCs in the original training group, i.e. the augmentation level. Thus a '+ (0)' symbol means that the original training group was used, with no augmentation.

Test Group	Optimal training group configuration 1	2	3	4	5	AUC with synthetic LCs	AUC w/o synthetic LCs	AUC original
1	+ (0)	−	−	−	−	0.991	0.991	0.993
2	+ (0)	+ (2)	−	−	−	0.989	0.990	0.988
3	−	+ (0)	+ (0)	+ (5)	+ (5)	0.968	0.958	0.926
4	−	+ (0)	+ (5)	+ (10)	+ (5)	0.919	0.887	0.791
5	−	+ (0)	+ (6)	+ (10)	+ (2)	0.842	0.709	0.636

Table 10. Optimal configuration for STACCATO classification along with their AUCs. For comparisons the AUC of the same configuration but without adding synthetic LCs (i.e., corresponding to setting all numbers in the parentheses to zero) and the AUC using the original $\mathcal{B}_{\text{train}}$, computed without STACCATO are also given. (The original analysis is the same as summarized in Figure 14. Each AUC is computed using only the corresponding generalization group, resulting in small variations in the last column compared with Figure 14). Notation is the same as in Table 9.

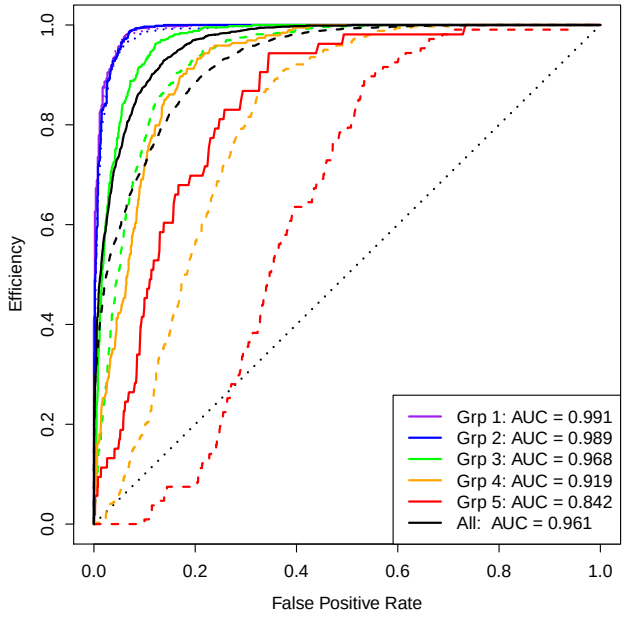

Figure 15. Classification with STACCATO. ROC curves of the optimal configurations given in Table 10. The curves are computed using the generalization groups. For comparison, ROC curves from Figure 14 (without STACCATO) are plotted as dashed curves.

(iv) Time-aligning and normalizing the LCs as described in Section 3;

(v) Separately computing diffusion maps for each of the augmented training groups, including the Nyström extension described in Appendix A2;

(vi) Classifying the SNe in each training group using a separate random forest as described in Section 4.2.

Step (iii) requires an augmentation scheme. In our current implementation each scheme is determined via recursive optimization over a validation group. Other schemes that do not require validation groups are possible and are the subject of ongoing research.

6.2.3 STACCATO results

Table 10 lists the optimal configurations (determined by optimizing the AUC on the validation groups), along with their AUCs, computed on the generalisation groups. For comparison we also give the AUCs of the same configuration but without adding the synthetic LCs to the augmented training groups. (The training groups are still augmented in that they may combine more than one of the original training groups.) We also report the AUCs using each of the generalization groups without any augmentation. (Although this analysis follows the same steps as that illustrated in Figure 14, the AUC values are slightly different than those reported in Figure 14 because the original analysis used the entire test set rather than just the generalization groups.) STACCATO delivers significantly larger AUC across all test groups, particularly when synthetic LCs are included in the augmented training groups.

Figure 15 compares the ROC curves for the optimal configurations given in Table 10 (i.e., using the augmented training groups with synthetic LCs) with those given in Figure 14. The substantial classification improvements apparent for Groups 3–5 are largely driven by the addition of synthetic LCs. The optimal training group for Group 5, the group that benefits the most from augmentation, includes 436 LCs, of which $\approx 38\%$ are synthetic. In contrast, Groups 1 and 2 show no improvement when synthetic LCs are introduced. Combining classification among all five groups, the AUC increases from 0.919 in the original model to 0.961 with STACCATO. Compared with the AUC of 0.977 obtained when using the unbiased training set, STACCATO performs almost as well as this gold standard.

Ultimately, we propose to use STACCATO to compute the probability that any particular photometrically observed SN is a SNIa. These probabilities can be used as weights or priors in secondary analyses and and their computation does not require artificial hard thresholding. For comparison with existing methods, however, we propose a dynamic threshold that increases with the propensity group number and thus accounts for the fact that there are more SNIa among the bright SNe (low numbered groups). Specifically, setting arbitrary classification thresholds of $\gamma_{\rm Ia} = 0.50, 0.60, 0.70, 0.80, 0.90$ for Groups 1–5, we obtain an efficiency of $e_{\rm Ia} = 0.846$, a purity of $p_{\rm Ia} = 0.834$ and an optimality criterion of $\zeta_{\rm Ia} = 0.529$ on the generalisation groups. Even without any optimization of the classification thresholds, we achieve a better optimality criterion than any of the previous results on the same data set (see Table 4). We therefore simply adopt our arbitrary choice for the classification thresholds, without any form of optimization on the threshold, to further compare STACCATO results, obtained both with $\mathcal{U}_{\rm train}$ and $\mathcal{B}_{\rm train}$, with other published methods in Table 4.

The comparison is complicated by the fact that different authors used different version of the SN classification challenge simulations. Richards et al. (2012) and Lochner et al. (2016) adopted an updated version of the simulated data with a number of corrections and bug fixes that was released after the challenge (Kessler et al. 2010b). Newling et al. (2011) report that classification using the updated simulation is more difficult. This might be in part due to the fact that two bugs where discovered in the original simulation (the one used in this paper), which made classification easier: some SNIa were too bright, while *all* non-SNIa were too dim by a factor $(1 + z)$ (Kessler et al. 2010b) (We will apply STACCATO to the newer, more difficult simulation in a future paper.) Further, the results of the original challenge are presented as a function of redshift, with no combined result reported, and only $\mathcal{B}_{\rm train}$ was considered. Kessler et al. (2010b, p. 8) note that the "most stable" classifiers obtained $\zeta_{\rm Ia} \in [0.3, 0.45]$ at all redshifts. (Some classifiers achieved $\zeta_{\rm Ia} = 0.6$ at some redshift, but with large downwards variations of a factor of 2 at other redshift values, suggesting an overall poorer performance once averaged over redshift.) Although none of the entries in Table 5 are exactly comparable, results obtained with STACCATO appear to be superior to all previously published methods, as evidenced by the its higher optimality criterion and AUC values.

7 DISCUSSION AND CONCLUSIONS

Augmenting a biased training set with synthetic LCs simulated from fitted GPs can dramatically improve SN classification, without needing to observationally obtain an expensive unbiased training set. To our knowledge, this is the first demonstration of this technique. Although the quality of the GP LC models vary when judged by eye, we find that classification results are nonetheless insensitive to the choice of GP covariance function.

We achieve increased performance levels in AUC, from 0.93 using the biased training set to 0.96 using STACCATO. This compares well with the best result in the literature, an AUC of 0.88, obtained by Lochner et al. (2016) with a biased training set. STACCATO also compares favourably with the gold standard of 0.98 obtained using our method with an unbiased training set, which itself matches the performance of previous methods applied with representative training sets (Lochner et al. 2016). We also obtain significantly higher purity values than previous studies with the same simulated data (0.59, compared to the previous best result of 0.55 obtained by Varughese et al. (2015)).

Although Möller et al. (2016) obtain an AUC of 0.98 by applying random forests and boosted decision trees to a simulated SNLS3 SN sample, this simulation is explicitly designed to include a larger number of fainter objects than the standard training set. This effectively creates a representative training set. Indeed they find that a set of specially designed selection cuts is "essential" for good classification. Our approach demonstrates that such cuts can be avoided.

Dai et al. (2017) also report an AUC of 0.98 on a set of LSST-like SN simulations using a random forest classifier. However, this result requires a representative training set that is 30% the size of the test set. As Dai et al. (2017) acknowledge, this is impossible to obtain in practice, hence they suggest using a simulation to generate a large representative training set. Again, STACCATO achieves comparable performance without needing such a large representative training set.

Another advantage of STACCATO is that the propensity scores groups can be used to assess the classification quality. In propensity scores groups with a small training set, accurate classification may not be possible, while it can be very accurate in groups with a large training set. Furthermore, STACCATO enables the identification of a subset of SNe for spectroscopic follow up with a high probability of containing a sizable proportion of SNIa. For example, among the 20 SNe in Group 5 that are most likely to be SNIa (as judged by random forest votes) five are actually SNIa, a proportion of 25%. Given that only 2.7% of the SNe in Group 5 are SNIa, this is a considerable improvement in the proportion of SNIa among those identified for follow-up. For comparison, when using the biased training set, only one SNIa is found among the 200 most likely SNe in the same group, a fraction of 0.5%. Thus STACCATO leads to a 50 fold improvement in the probability that a SN identified for spectroscopic follow-up at high redshift is a SNIa (i.e., the true positive rate).

The propensity score groups have differing proportions of SNIa even after controlling for redshift and brightness, which suggests that there are other variables that could be informative in modeling the inclusion of SNe in the training set (e.g., via logistic regression). It would be interesting to include additional variables (such as the signal to noice ratio, data quality cuts, etc) that bias spectroscopic follow-up in favour of SNIa. This would enable the propensity scores to identify groups where the training set better represents the test set in terms of the distributions of the additional variables in the training and test sets.

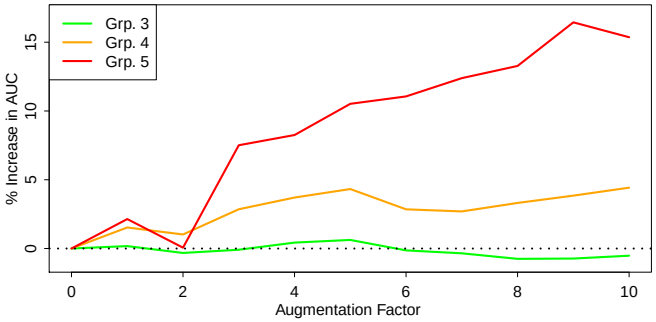

Figure 16. Percent increase in AUC as a function of the augmentation factor. All groups are trained on the combined observed training sets from Groups 2–5 using a common augmentation factor in Groups 3–5, as plotted on horizontal axis. Zero corresponds to no augmentation and is the baseline against which increasing degrees of augmentation are compared.

The current implementation of STACCATO requires the SN types for the validation groups in order to optimize the configuration of the augmented training groups. In a real application, the SN types for the test groups are unknown, and therefore the validation groups are not available. Simulation studies, however, could be used to optimize the configuration of the training sets before being rolled out on the actual test set. However, even in the absence of a definitive scheme for optimizing the augmentation scheme, Figure 16 demonstrates that augmenting small training groups is almost always beneficial. The figure depicts the effect of augmentation on AUC in Groups 3–5. All three groups are trained using (the original) Training Groups 2–5 and using the same augmentation factor (from 0 to 10) in Groups 3–5. Group 3 is relatively insensitive to augmentation; this is also indicated in Table 10. Groups 4 and 5 both benefit from increasing levels of augmentation. In particular, the trend of the AUC as the augmentation factor increases is either positive or flat, with only local minor decrements. Thus, an augmentation factor of up to ten seems to be a safe choice for small training groups, even without optimizing the augmentation scheme using a validation group.

We leave further investigation of both the tuning of STACCATO in the absence of a validation group and accounting for errors in or unavailable redshift measurements to future work.

Acknowledgements: We thank David Stenning for helpful suggestions on the statistical methods developed in this paper and Bruce Bassett, Michelle Lochner and Rick Kessler for useful comments on a draft. This work was supported by Grant ST/N000838/1 from the Science and Technology Facilities Council (UK). RT was partially supported by an EPSRC "Pathways to Impact" grant. RT and DvD were supported by a Marie-Skodowska-Curie RISE (H2020-MSCA-

RISE-2015-691164) Grant provided by the European Commission.

References

Adler R. J., 1990, An introduction to continuity, extrema, and related topics for general Gaussian processes. Hayward, CA. Institute of Mathematical Statistics

Budavari T., 2009, Astrophys. J., 695, 747

Byrd R. H., Lu P., Nocedal J., Zhu C., 1995, SIAM Journal on Scientific Computing, 16, 1190

Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P., 2002, Journal of artificial intelligence research, 16, 321

Coifman R. R., Lafon S., 2006, Applied and computational harmonic analysis, 21, 5

Dai M., Kuhlmann S., Wang Y., Kovacs E., 2017

Eddington A. S., 1913, MNRAS, 73, 359

Freeman P., Newman J., Lee A., Richards J., Schafer C., 2009, Monthly Notices of the Royal Astronomical Society, 398, 2012

Friedman J., Hastie T., Tibshirani R., 2009, The elements of statistical learning, 2 edn. Springer series in statistics Springer, Berlin

Gelman A., Carlin J. B., Stern H. S., Rubin D. B., 2013, Bayesian data analysis, third edn. Taylor & Francis

Gibbs M. N., 1997, PhD thesis, University of Cambridge

Givens G. H., Hoeting J. A., 2012, Computational statistics. John Wiley & Sons

Ha T. M., Bunke H., 1997, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19, 535

Hlozek R., et al., 2012, Astrophys. J., 752, 79

Hoyle B., Rau M. M., Bonnett C., Seitz S., Weller J., 2015, Mon. Not. Roy. Astron. Soc., 450, 305

Japkowicz N., et al., 2000, in AAAI workshop on learning from imbalanced data sets. pp 10–15

Kelly B. C., 2007, Astrophys. J., 665, 1489

Kessler R., Scolnic D., 2017, Astrophys. J., 836, 56

Kessler R., Conley A., Jha S., Kuhlmann S., 2010a, arXiv preprint arXiv:1001.5210

Kessler R., et al., 2010b, Publications of the Astronomical Society of the Pacific, 122, 1415

Kim A. G., et al., 2013, Astrophys. J., 766, 84

Knights M., Bassett B. A., Varughese M., Hlozek R., Kunz M., Smith M., Newling J., 2013, JCAP, 1301, 039

Kunz M., Bassett B. A., Hlozek R., 2007, Phys. Rev., D75, 103508

LSST Science Collaboration et al., 2009, preprint, (arXiv:0912.0201)

Lafon S., Lee A. B., 2006, IEEE transactions on pattern analysis and machine intelligence, 28, 1393

Lochner M., McEwen J. D., Peiris H. V., Lahav O., Winter M. K., 2016

Malmquist K. G., 1925, Meddelanden fran Lunds Astronomiska Observatorium Serie I, 106, 1

Möller A., et al., 2016, JCAP, 1612, 008

Newling J., et al., 2011, 414, 1987

Perlmutter S., et al., 1999, The Astrophysical Journal, 517, 565

Rasmussen C. E., Williams C. K. I., 2006, Gaussian Processes for Machine Learning. The MIT Press

Richards J. W., Homrighausen D., Freeman P. E., Schafer C. M., Poznanski D., 2012, Monthly Notices of the Royal Astronomical Society, 419, 1121

Riess A. G., et al., 1998, The Astronomical Journal, 116, 1009

Roberts S., Osborne M., Ebden M., Reece S., Gibson N., Aigrain S., 2012, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 371

Rosenbaum P. R., Rubin D. B., 1984, Journal of the American statistical Association, 79, 516

Rubin D., et al., 2015, Astrophys. J., 813, 137

Seikel M., Clarkson C., Smith M., 2012, Journal of Cosmology and Astroparticle Physics, 2012, 036

Varughese M. M., von Sachs R., Stephanou M., Bassett B. A., 2015, MNRAS, 453, 2848

APPENDIX A: DIFFUSION MAPS

A1 Construction of the Diffusion Maps

Let $\mathcal{Y}_c = \{Y_i, \ldots, Y_N\}$ denote the normalised LC data in colour band c and define a metric $d_c : \mathcal{Y}_c \times \mathcal{Y}_c \mapsto [0, \infty)$. The specific choice of d_c is introduced in Section A3. The metric is used to construct a Markov chain. The LCs form the state space and the probabilities of the Markov chain jumping from one LC to another are determined by the distance between the LCs under the metric, d_c. Similar LCs are given high transition probabilities. The pairwise distances are transformed with a Gaussian kernel and normalised so that the transition probabilities sum to one. Define the 'weight' function, $w_c : \mathcal{Y}_c \times \mathcal{Y}_c \mapsto (0, 1]$, and the transition probabilities $p_c(x, y)$ as

$$w_c(x, y) = \exp\left(-\frac{d_c(x, y)^2}{\varepsilon_c}\right) \tag{A1}$$

$$p_c(x, y) = \frac{w_c(x, y)}{\sum_z w_c(x, z)}, \tag{A2}$$

where $\varepsilon_c > 0$ is a tuning parameter. Note that by applying (A1) a distortion favouring 'local jumps' in the Markov chain is introduced due to the exponential decay in (A1). As a result, for many LC pairs w_c is small. For computational efficiency a sparsity parameter δ is introduced, such that if $w_c(\cdot, \cdot) < \delta$ then it is set to zero; we use $\delta = 10^{-5}$. With these definitions we can compute a sparse $N \times N$ transition matrix P_c giving the probabilities of jumping between all pairs of c-band LC.

Define the diffusion distance between two LCs from the transition probabilities as

$$D_c(x, y)^2 = \sum_{z \in \mathcal{Y}_c} \frac{[p_c(x, z) - p_c(y, z)]^2}{\phi_{0c}(z)}, \tag{A3}$$

(notice that z here is a dummy index, not redshift) where $\phi_{0c}(z)$ is the stationary distribution of the Markov chain,

$$\phi_{0c}(x) = \frac{\sum_z w_c(x, z)}{\sum_y \sum_z w_c(y, z)}. \tag{A4}$$

Assuming that the Markov chain is irreducible, it follows from the construction of the chain that the stationary distribution exists and is unique (Coifman & Lafon 2006). The diffusion distance in (A3) measures the similarity between two LCs as the squared difference in probabilities of transitioning from x to z versus y to z in a weighted sum over all $z \in \mathcal{Y}_c$.

The diffusion distances can be calculated from the right eigenvalues and eigenvectors of the transition matrix P_c, specifically,

$$D_c(x, y)^2 = \sum_{j=1}^{N-1} \lambda_j^2 \left(\psi_j(x) - \psi_j(y)\right)^2 \tag{A5}$$

$$\approx \sum_{j=1}^{m} \lambda_j^2 \left(\psi_j(x) - \psi_j(y)\right)^2, \tag{A6}$$

where $|\lambda_0| \geqslant |\lambda_1| \geqslant \cdots \geqslant |\lambda_{N-1}|$ are the eigenvalues sorted in descending order and ψ_j are the corresponding eigenvectors normalised according to ϕ_{0c}, such that $\sum_x \psi_j^2(x)\phi_{0c}(x) = 1$, with $\psi_j(x)$ equal to entry x of eigenvector j. (To simplify notation we suppress the subscript c indicating the colour band for ψ and λ.) The first eigenvector and eigenvalue has been left out of the sum in (A5). This is because ψ_0 can be shown to be constant in its entries. Because the eigenvalues are in decreasing order, most of the diffusion distance can be explained by the first $m \ll N$ terms in the sum, hence the approximation in (A6). In our numerical results, we set m to be the smallest value such that $\lambda_m < 0.05\lambda_1$ (in (A6)), with a maximum cap of 25.

Using (A6) the diffusion map, $\mathbf{\Psi}_c : \mathcal{Y}_c \mapsto \mathbb{R}^m$, is defined

$$\mathbf{\Psi}_c : x \mapsto [\lambda_1\psi_1(x), \lambda_2\psi_2(x), \ldots, \lambda_m\psi_m(x)]\,. \tag{A7}$$

From this mapping the approximate diffusion distances (A6) can be calculated as the Euclidean distance between the representation of the LCs in the diffusion space. For further details on diffusion maps, see e.g. Lafon & Lee (2006) and Coifman & Lafon (2006).

A2 The Nyström Extension

Richards et al. (2012) apply the diffusion map to the entire dataset, including training and test sets. This approach results in what is called a 'semi-supervised' classification algorithm, in that the unlabelled LCs are used to construct the representation of the LCs in $\mathbb{R}^m$. In contrast, we use only the training data to construct the diffusion map. This has two benefits. First, computing and storing the distance matrix P_c requires considerable computational resources and scales as $\mathcal{O}(N^2)$ where N is the number of LCs. Hence computing the diffusion map on only about 7% of the data is advantageous from a practical perspective. Second, in Section 6 we show that computing the diffusion map on the training data only increases classification performance. However, this approach requires a method to map new LCs (i.e. the curves that need to be classified and were not included in the training set) into the diffusion space. The Nyström extension gives a simple method to do this.

The following outline of the Nyström extension is based on Freeman et al. (2009). Suppose there are n c-band LCs in the training set, and k unlabelled c-band LCs in the test set. The first step of the Nyström extension is to compute a $k \times n$ matrix W_c with entries equal to the normalized $w_c(x, y)$ with x representing a LC in the test set and y a LC in the training set. Using the same ε_c parameter, the rows of W_c are normalised, as with P_c, following (A2).

The eigenvectors used to construct the diffusion map in (A7) can be arranged in a $n \times m$ matrix Ψ_c. The corresponding eigenvalues are used to construct an $m \times m$ diagonal matrix Λ_c with diagonal entry i equal to $1/\lambda_i$. Finally, the diffusion-space coordinates of the test set LCs are given by the rows of the $k \times m$ matrix built as follows:

$$\Psi_c' = W_c \Psi_c \Lambda_c. \tag{A8}$$

A3 The Light Curve Metric

The mean squared difference,

$$d_c(x, y) = \frac{1}{\tilde{t}_u^c - \tilde{t}_l^c + 1} \sum_{t=\tilde{t}_l^c}^{\tilde{t}_u^c} \left(\tilde{f}_x^c(t) - \tilde{f}_y^c(t)\right)^2, \quad x, y \in \mathcal{Y}, \tag{A9}$$

is used as the LC metric, where $\tilde{t}_l^c = \max\left(\tilde{t}_{x,1}^c, \tilde{t}_{y,1}^c\right)$ and $\tilde{t}_u^c = \min\left(\tilde{t}_{x,n_x^c}^c, \tilde{t}_{y,n_y^c}^c\right)$ form the lower and upper bounds of the common time domain of the two LCs. When $\tilde{t}_u^c < \tilde{t}_l^c$, i.e., when there is no common domain for the two LCs, $d_c(x, y)$ is set to one. (This is a large value relative to the scales of $\tilde{f}$.) Note that this does not necessary result in the two LCs being far apart in the diffusion space following (A3).