SUBMITTED TO APJ
Preprint typeset using LaTeX style emulateapj v. 01/23/15

MODELING THE *Swift* BAT TRIGGER ALGORITHM WITH MACHINE LEARNING

PHILIP B. GRAFF
Department of Physics and Joint Space Science Institute, University of Maryland, College Park, MD 20742, USA and
NASA Goddard Space Flight Center, 8800 Greenbelt Rd., Greenbelt, MD 20771, USA

AMY Y. LIEN
NASA Goddard Space Flight Center, 8800 Greenbelt Rd., Greenbelt, MD 20771, USA

JOHN G. BAKER
NASA Goddard Space Flight Center, 8800 Greenbelt Rd., Greenbelt, MD 20771, USA

TAKANORI SAKAMOTO
Department of Physics and Mathematics, College of Science and Engineering, Aoyama Gakuin University, 5-10-1 Fuchinobe, Chuo-ku, Sagamihara-shi, Kanagawa 252-5258, Japan

Submitted to ApJ

ABSTRACT

To draw inferences about gamma-ray burst (GRB) source populations based on *Swift* observations, it is essential to understand the detection efficiency of the *Swift* burst alert telescope (BAT). This study considers the problem of modeling the *Swift* BAT triggering algorithm for long GRBs, a computationally expensive procedure, and models it using machine learning algorithms. A large sample of simulated GRBs from Lien et al. (2014) is used to train various models: random forests, boosted decision trees (with AdaBoost), support vector machines, and artificial neural networks. The best models have accuracies of $\gtrsim 97\%$ ($\lesssim 3\%$ error), which is a significant improvement on a cut in GRB flux which has an accuracy of 89.6% (10.4% error). These models are then used to measure the detection efficiency of *Swift* as a function of redshift z, which is used to perform Bayesian parameter estimation on the GRB rate distribution. We find a local GRB rate density of $n_0 \sim 0.48^{+0.41}_{-0.23}$ Gpc^{-3}yr^{-1} with power-law indices of $n_1 \sim 1.7^{+0.6}_{-0.5}$ and $n_2 \sim -5.9^{+5.7}_{-0.1}$ for GRBs above and below a break point of $z_1 \sim 6.8^{+2.8}_{-3.2}$. This methodology is able to improve upon earlier studies by more accurately modeling *Swift* detection and using this for fully Bayesian model fitting. The code used in this is analysis is publicly available online[a].

Keywords: gamma rays: general, methods: data analysis

1. INTRODUCTION

Long gamma-ray bursts (GRBs) are related to core-collapse supernovae from the death of massive stars. These are important for studying star-formation history, particularly in the early universe where other methods become difficult. The *Swift* space telescope (Gehrels et al. 2004) is able to detect and localize these out to large distances and quickly downlink the data to the ground. These abilities enable prompt ground-based followup observations that can provide redshift measurements of the GRBs. To date, *Swift* has detected over 900 GRBs, of which $\sim 30\%$ have redshift measurements. From these observations, one can try to infer the intrinsic GRB rate that is connected to stellar evolution over the history of the Universe. Many researchers have used *Swift*'s observations to study intrinsic GRB redshift and luminosity distributions, and the implications for star-formation history (e.g., Guetta and Della Valle 2007; Guetta and Piran 2007; Yüksel et al. 2008; Kistler et al. 2008; Butler et al. 2010; Robertson and Ellis 2012; Pélangeon et al. 2008; Salvaterra et al. 2009; Campisi et al. 2010; Wanderman and Piran 2010; Virgili et al. 2011; Qin et al. 2010; Salvaterra et al. 2012; Coward et al. 2013; Kanaan and de Freitas Pacheco 2013; Wang 2013; Lien et al. 2014; Howell et al. 2014; Yu et al. 2015; Petrosian et al. 2015; Pescalli et al. 2015).

Several studies have suggested that the GRB rate at high redshift ($z \gtrsim 5$) is larger than the expectation based on star-formation rate (SFR) measurements (e.g., Le and Dermer 2007; Yüksel et al. 2008; Kistler et al. 2009; Butler et al. 2010; Ishida et al. 2011; Tanvir et al. 2012; Jakobsson et al. 2012; Lien et al. 2014). This result could imply several possibilites, such as a larger star-formation rate in the early universe(e.g., Kistler et al. 2009; Tanvir et al. 2012), an evolving luminosity function(e.g., Virgili et al. 2011; Pescalli et al. 2015), or a different GRB to supernova ratio (i.e. a different scenario of stellar evolution) due to a different environment in the early universe (e.g., Woosley and Heger 2012).

However, it remains difficult to constrain the GRB rate. Though *Swift* has observed a large population of GRBs only some of these have measured redshifts. Even with a relatively complete redshift sub-sample, there are complicated selection effects from the complex trigger algorithm adopted by the burst alert telescope (BAT) on-board *Swift* and the difficulty in searching through a large parameter space. It is challenging to distinguish the luminosity function and the redshift distribution using the observational data. We address some of these issues with a machine learning approach to produce a

pgraff@umd.edu
amy.y.lien@nasa.gov
john.g.baker@nasa.gov
[a] https://github.com/PBGraff/SwiftGRB_PEanalysis

fast, but reliable, treatment of *Swift*'s instrumental selection effects, thereby enabling a robust Bayesian treatment of population model analysis.

Machine learning (ML) is a field of research that involves designing algorithms (MLAs) for building models that learn from generic data. The models are fit to a set of training data in order to make predictions or decisions. Often, the original training data come from actual observations or simulations of a complex process. The models trained by MLAs can be evaluated very quickly for any new example after a one-time cost of training the model.

In this study, we look to aid the analysis of GRB data by using MLAs to train models that emulate the *Swift* trigger algorithm. Our training data comes from simulations of GRB populations computed by Lien et al. (2014).

The structure of this paper is as follows. In Section 2 we describe the aspects of *Swift* and its model for triggering on incident GRBs that are relevant to GRB population inferences. Then, in Section 3 we describe the machine learning algorithms used and compared in this study. Section 4 presents the results of training the different ML models on the training data from the *Swift* pipeline. We apply a trained ML model for accelerating Bayesian inference with faster likelihoods in Section 5, fitting the parameters of the intrinsic GRB rate distribution. Section 6 compares our study to previous work estimating the intrinsic distributions of long GRBs with *Swift* observations. Lastly, in Sections 7 and 8 we summarize and propose future projects to follow-up.

2. THE *Swift* DETECTION ALGORITHM

The burst alert telescope (BAT) on-board Swift adopts over 500 rate trigger criteria based on photon count rate in the raw light curve. Moreover, the burst needs to pass the "image threshold" determined by the signal-to-noise ratio estimated from an image generated on-board based on the duration found by the rate trigger criteria. Each rate trigger criterion uses a different energy band, different part of the detector plane, and different foreground and background durations for calculating the signal-to-noise ratio. In addition to the rate trigger algorithm, the BAT also generates an image every $\gtrsim$ minute to search for bursts that are missed by the rate trigger method (which is the so-called "image trigger") (Barthelmy et al. 2005; Fenimore et al. 2003, 2004; McLean et al. 2004; Palmer et al. 2004).

This complex trigger algorithm successfully increases the number of GRB detections. However, it also increases the difficulty of estimating the detection theshold, which is curcial for probing many intrinsic GRB properties from the observations. To address this problem, Lien et al. (2014) developed a code that simulates the BAT trigger algorithm, and used it to study the instrinsic GRB rate and luminosity function. This "trigger simulator" follows the same trigger algorithm and criteria for the rate trigger as those adopted by the BAT, and mimics the image threshold and image trigger (see Lien et al. (2014) for detailed descriptions). Although the trigger simulator can be used to address the complex detection thresholds of the BAT, it takes $\sim$ 10 seconds to a few minutes to simulate the trigger status of a burst using a common PC with the 2.7 GHz Intel Core processor (the speed mainly depends on the number of bins in the light curve). Therefore, it is computationally intensive to perform a large number of simulations to cover a wide parameter space. This is where machine learning is able to accelerate our analysis.

3. MACHINE LEARNING ALGORITHMS

To generate a fast emulator for the *Swift* trigger simulator, we consider a variety of supervised learning algorithms, where the goal is to infer a function from labeled training data. Each example consists of input properties which are used to predict the output label.

Here we briefly describe each of the machine learning algorithms used in this study. We denote the set of input features by $\boldsymbol{x}$ and the machine learning model's predicted output is given by $y(\boldsymbol{x})$. The inputs are a set of 15 parameters describing the GRB and detector as detailed in Table 1. Depending on the MLA, the output may be a discrete label, e.g. $\{0, 1\}$, or it may be a continuous probability in $[0, 1]$ and is designed to be the probability that a GRB, as specified by the features in $\boldsymbol{x}$, is detected by *Swift*'s BAT. The true output is given by $\boldsymbol{t}$ and is 0 for a non-detection and 1 for a detection.

3.1. *Random Forests and AdaBoost*

Random forests and AdaBoost both involve creating ensembles of decision trees, so we first introduce these as a machine learning model. In a decision tree, binary splits are performed on the training data input features, the dimensions of $\boldsymbol{x}$. In training a tree on data, a series of splits are made that choose a dimension and a threshold that optimize some criterion. Examples of this criterion are the accuracy of the resulting classifications (maximize; equivalently minimize errors) or the Gini impurity (minimize) which given by

$$G = 1 - \sum_{i=\{0,1\}} f_i^2, \tag{1}$$

where f_i is the fraction of correctly classified samples labeled with value i. This measure aims to make each sub-set resulting from a branch as "pure" as possible in the class labels of its members. Each split creates a pair of "branches", one with each class label. These splits are made until a stopping condition is reached (e.g. the samples are all of uniform class, a maximum number of splits has been reached, or the number of samples left to split among has fallen below a minimum value). This branch now becomes a "leaf" that assigns a class to all samples ending there. When a new event of unknown class is put into the tree, the tree will pass it through the learned splits/branches until it reaches a leaf, at which point it will be labeled according to the label of the leaf. An example tree fit to this data (with a hard limit of 3 in depth) is shown in Figure 1; it has a classification accuracy of 93.9% on the training data to which it was fit. Trees fit in the later models will be much larger and thus more accurate.

3.1.1. *Random Forests*

Random forests (RFs) (Breiman 2001) improve upon classical decision trees by training an ensemble of trees that vote on the final classification. A "strong learner" (the RF) is created from an ensemble of "weak learners" (decision trees). In a RF, many decision trees are trained on the data – often hundreds. To obtain many different trees, at each split in a tree, a random subset of the dimensions of $\boldsymbol{x}$ are chosen and the optimal binary split to be made out of these dimensions is made. Furthermore, each tree is trained on a bootstrap sample of the data; the original K points are sampled with replacement to form a new set of K points that may contain repeats. A RF thus guards against overfitting to the training data and potentially badly performing individual trees. A single tree

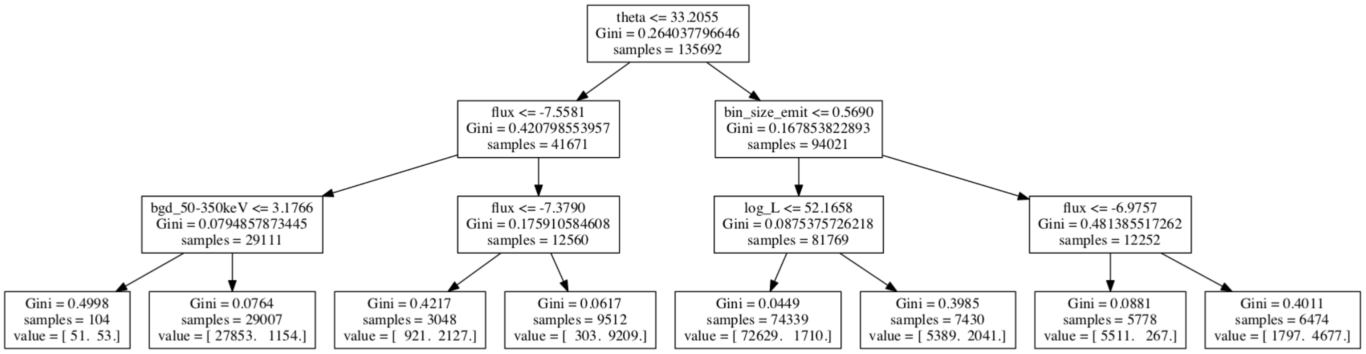

Figure 1. A decision tree fit to the *Swift* training data with a maximum depth of 3. An accuracy of 93.9% is achieved. Each box shows the parameter chosen for branching and the threshold used, as well as the total number of samples used to make that decision. The Gini factor shown is that from Equation (1) for the subset at that location. At the leaves, the number of items with class 0 and class 1 are shown; the tree can assign class probabilities based on this split at the leaf that any new sample arrives at after following the branches down.

can provide a probabilistic classification, $y_{\rm DT}(\boldsymbol{x}) \in [0,1]$, and combining many allows us to obtain a near-continuous probability, $y_{\rm RF}(\boldsymbol{x}) \in [0,1]$ by using

$$y_{\rm RF}(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} y_{{\rm DT},n}(\boldsymbol{x}), \tag{2}$$

with N being the number of trees in the forest. This value we obtain as $y_{\rm RF}(\boldsymbol{x})$ is simply the probability that the GRB described by $\boldsymbol{x}$ is detected by *Swift*.

We use the implementation of RFs in the `scikit-learn`[1] Python library (Pedregosa et al. 2011).

3.1.2. *AdaBoost*

AdaBoost is short for "Adaptive Boosting", a meta-algorithm for machine learning (Freund and Schapire 1997). It creates a single strong learner from an ensemble of weak learners, much like RFs. However, in the boosting framework, the decision trees are trained iteratively and when added together (as in Equation (2)) are weighted, typically based on their accuracy. Additionally, unlike for RFs, the training examples will not be all equally weighted when evaluating the accuracy. After an individual decision tree is added to the ensemble, the training data is reweighed so that examples that are misclassified increase in weighting and those classified correctly decrease in weighting. Therefore, future decision trees will attempt to better fit examples previously misclassified. In this way, the overall ensemble prediction may become more accurate.

Boosting may be applied to any machine learning algorithm, but in this work we apply it only to the decision tree weak learner (the other classifiers qualify as strong learners on their own and would thus likely not benefit significantly from boosting). We use the implementation of AdaBoost for decision tree classifiers in `scikit-learn`. We note that the predicted probability, $y_{\rm AB}(\boldsymbol{x}) \in [0,1]$, is approximately continuous, similarly to $y_{\rm RF}(\boldsymbol{x})$.

3.2. *Support Vector Machines*

Support vector machines (SVMs) (Cortes and Vapnik 1995) are a tool for binary classification that finds the optimal hyper-plane for separating the two classes of training samples. Events are classified by which side of the hyper-plane they fall on. The hyper-plane that maximizes the separation from points in either class will (in general) have minimal generalization error for new data points.

In a linear SVM, we label the two classes with $t_i \in \{-1,1\}$ corresponding to an un-detected GRB and a detected GRB, respectively. A hyper-plane separating the two classes will satisfy $\boldsymbol{w}\cdot\boldsymbol{x} - b = 0$, where $\boldsymbol{w}$ and b must be found by training on the data $\{\boldsymbol{x}\}$. If the classes are separable, we can place two parallel hyper-planes that separate the points and have no points between them in the "margin". This can be seen for a toy example in Figure 2. We describe these hyper-planes mathematically as

$$\boldsymbol{w}\cdot\boldsymbol{x} - b = \pm 1, \tag{3}$$

Examples will lie on either side of the two planes such that

$$t_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i - b\right) \geq 1 \tag{4}$$

for all samples, $\boldsymbol{x}_i$. As the samples are typically not separable, we introduce slack variables $\xi_i \geq 0$ that measure the misclassification of $\boldsymbol{x}_i$ by setting

$$t_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i - b\right) \geq 1 - \xi_i\,. \tag{5}$$

We then seek to minimize

$$\text{Cost}(\boldsymbol{w},\boldsymbol{\xi},b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i \tag{6}$$

subject to the constraint in Equation 5. The C parameter is a penalty factor for misclassification and this optimization will face the trade-off between a smaller margin and smaller misclassification error. The cost function seeks to maximize the distance between the two hyper-planes at the margin edges, which is given by $2/||\boldsymbol{w}||$. This separation by hyper-plane is demonstrated for a toy example in Figure 2.

The two classes of points are generally not easily separated in the original parameter space of the problem. Therefore, we map the points into a higher-dimensional space where they may be more easily separated. To make this a computationally tractable problem, we consider mappings such that the dot product between pairs of points may be easily computed in terms of the original variables by a kernel function, $k(\boldsymbol{x}_i,\boldsymbol{x}_j)$. Hyper-planes in the higher-dimensional space are defined as surfaces on which the kernel is constant. If the kernel is defined such that $k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ decreases as the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ move away from one another, then the kernel is a measure of closeness. Thus, the sum of many kernels like this can be used to measure the proximity of a sample data point to data points in the two classes; this distance can then be used to classify the

[1] `http://scikit-learn.org/stable/index.html`

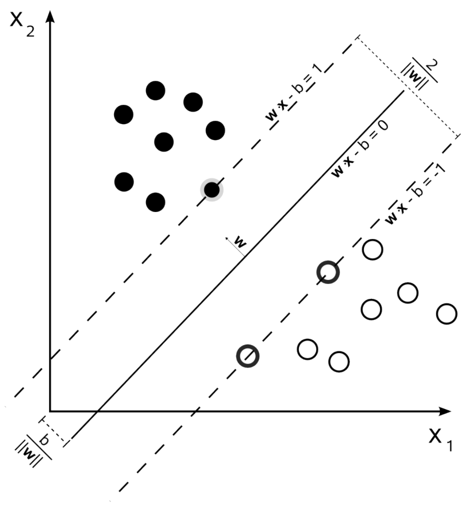

Figure 2. The maximum separating hyper-plane and the margin hyper-planes for a toy data set. The "support vectors" are the highlighted points along the margin hyper-planes. Image courtesy of Wikimedia Commons (Commons 2008).

point into one class or the other. This mapping can result in a very convoluted hyper-plane separating the two sets of points – this can accurately model the true classification boundary, but we must be careful not to overfit this to the training data.

In order to perform a non-linear separation, we employ a Gaussian kernel function (a.k.a. radial basis function),

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right) \tag{7}$$

where γ is a tunable parameter reflecting the width of the Gaussian.

A point is classified by which side of the learned hyper-plane it falls on, as determined (in our notation) by

$$y(\boldsymbol{x}) = \text{sign}\left(K(\boldsymbol{w}, \boldsymbol{x}) - b\right) , \tag{8}$$

where K is the aggregate kernel function that is a linear combination of the individual kernel functions (closeness to each of the other points). Minimizing the cost given by Equation (6) under the constraint of Equation (5) can be solved as a quadratic programming problem with the solution locally independent of all but a few data points, the "support vectors" of the model. These will be those samples closest to or on the margin in both classes and a weighted sum of distances from them will determine which class a new sample is in. Points that are not support vectors will have small or zero weight in the aggregate kernel function.

In this study, we use the implementation of SVMs in `scikit-learn`. A radial basis function is chosen and we perform 5-fold cross-validation to optimize the hyper-parameters of the model, γ and C. The model is also trained to allow for the prediction of continuous class probabilities, $y_{\text{SVM}} \in [0, 1]$[2].

3.3. *Artificial Neural Networks*

Artificial neural networks are a machine learning method that is inspired by the function of a brain. A neural network (NN) consists of interconnected nodes, each of which processes information that it receives and passes this product on to other nodes via weighted connections. In a feed-forward NN, these nodes are organized into layers that pass information uniformly in a certain direction. The input layer passes information to an output layer via zero, one, or many "hidden" layers in between. Each node in the network performs a simple function, but their combined activity can model complex relationships. A useful introduction to NNs as well as their training and use can be found in MacKay (2003).

A single node takes an input vector of activations $\boldsymbol{a} \in \Re^N$ and maps it to a scalar output $f(\boldsymbol{a}; \boldsymbol{w}, b)$ through

$$f(\boldsymbol{a}; \boldsymbol{w}, b) = g\left(b + \sum_{i=1}^{N} w_i a_i\right) , \tag{9}$$

where $\boldsymbol{w}$ and b are the parameters of the node, called the "weights" and "bias", respectively. The function, g, is the activation function of the node; we use the sigmoid, linear, and rectified linear activation functions in this work.

$$g(z) = \begin{cases} (1 + \mathrm{e}^{-z})^{-1} & \text{sigmoid} \\ z & \text{linear} \\ \max\{0, z\} & \text{rectified linear} \end{cases} \tag{10}$$

The sigmoid and rectified linear activations are used for hidden layer nodes and the linear activation is used for the output layer nodes to obtain values in $(-\infty, \infty)$. This is then converted into a probability by the softmax transform given by

$$y_j(\boldsymbol{x}; \boldsymbol{w}, b) \rightarrow \frac{\exp\left(y_j(\boldsymbol{x}; \boldsymbol{w}, b)\right)}{\sum_{l=\{0,1\}} \exp\left(y_l(\boldsymbol{x}; \boldsymbol{w}, b)\right)} \tag{11}$$

where j indexes over the output nodes. After the softmax, all output values are in $(0, 1)$ and sum to 1. We show here the case where there are only two output nodes for a binary classification problem; these values are degenerate, but the setup of one output node per class generalizes to the multi-class problem.

The weights and biases of all nodes in the network are the parameters that must be optimized with respect to the training data. The number of input nodes is the number of features given by the data. The two output nodes are the values in which are the probabilities that the input GRB features would result in detection or non-detection. In this work, we will take $y_{\text{NN}}(\boldsymbol{x})$ to be the continuous probability given in the output node for the "detection" class. Thus, the output is the predicted probability that the given input GRB features correspond to a detected GRB.

The optimization algorithm seeks to minimize the cross-entropy of the predicted probabilities, given by

$$\text{Cost}(\boldsymbol{p}) = -\sum_{i} \sum_{k=\{0,1\}} t_{i,k} \log y_k(\boldsymbol{x}_i) \tag{12}$$

where $\boldsymbol{p}$ is a parameter vector containing all of the weights and biases of the nodes in the NN. The index i is over all data samples in the training set and the index k is over the 2 output nodes corresponding to the non-detection and detection classes, respectively. $t_i = \{1, 0\}$ for a non-detection and $t_i = \{0, 1\}$ for a detection. This cost function pushes predicted probabilities toward their correct values with large penalties for incorrect predictions and is based in information theory. We take the value from the output node corresponding to the detection class as the probability that the input GRB, $\boldsymbol{x}$, is detected by *Swift*, $y_{\text{NN}}(\boldsymbol{x})$.

We use the SkyNet[3] algorithm (Graff et al. 2014) for training of the NN and refer the reader to that paper for more in-

[2] See the `scikit-learn` documentation for details on this procedure.

[3] `http://www.mrao.cam.ac.uk/software/skynet/`

formation on NNs, including the optimization function used, how the optimization is performed, and additional data processing that is performed. SKYNET provides an easy-to-use interface for training as well as an algorithm that will efficiently and consistently find the best fit NN parameters for the training data provided.

3.4. *Heuristics Used*

For each model's optimal settings, we compute the accuracy of predictions using a naïve probability threshold of 0.5 for the output probability for the detection class; i.e. $y_m(\boldsymbol{x})$ for the different models, m. This is later found to be close to optimal. We also plot the receiver operating characteristic (ROC) curves for the classifiers as seen in Figure 9 later. A ROC curve plots the true positive rate (a.k.a. recall) against the false positive rate. The F1-score is a useful metric for finding the optimal probability threshold to balance type I (false positive) and type II (false negative) errors. The F1-score takes values in $[0, 1]$ and is maximized at the optimal probability threshold. These values are given by:

$$\begin{aligned}
\mathrm{TP} &= \#\text{ positives correctly labeled} \\
\mathrm{TN} &= \#\text{ of negatives correctly labeled} \\
\mathrm{FP} &= \#\text{ of negatives labeled as positive} \\
\mathrm{FN} &= \#\text{ of positives labeled as negative} \\
\mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = \text{recall} \\
\mathrm{FPR} &= \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} \\
\text{precision} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\
\text{F1-score} &= 2\,\frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}
\end{aligned}$$

where positives are detections and negatives are non-detections of GRBs. For a random classifier, the ROC will be a diagonal line from $(0, 0)$ to $(1, 1)$. Better classifiers will be above and to the left of this line. A common measure is the area under the curve (AUC), which is the integrated area under the ROC. Values closer to 1 indicate better classifiers.

In this study, we will use the ROC (with AUC) to find the MLA that best models the *Swift* pipeline. We can then use the F1-score to identify the best probability threshold for declaring a detection from the predictions of the model.

3.5. *Cross-Validation*

To measure the performance for each type of MLA, we perform hyper-parameter optimization over a range of settings for each. To properly compare these settings against each other, we perform *cross-validation*. In this setup, with 5 folds, we split the data into 5 random subsets of equal size. In training, we train 5 models for each setting, using 4 of the 5 during fitting and then evaluating the model on the left-out set. Thus we make predictions on the entire set but without having trained the model on the data it was predicting (this would lead to over-fitting).

Once the optimal model settings are found for each MLA, the entire data set is used to re-train a model with those values. This model is then evaluated on the left-out validation data set so as to compare it with the other MLAs. This latter test is much more stringent, as the evaluation data is from different populations than what was used in training. This better reflects how the ML model will be used in practice and is used to pick a MLA and model fit for use in Bayesian parameter estimation (Section 5).

4. MACHINE LEARNING RESULTS

In this section we present the details of the MLA model fitting we performed. We describe the data set used for training and validation followed by results from hyper-parameter optimization searches performed for each classifier. The hyper-parameter optimization uses only the training data and evaluates different settings with cross-validation as described in Section 3.5. Once we obtain optimal settings for each MLA, we evaluate the models on a validation data set (separate from the training data) for final performance measurement and comparison.

4.1. *Training Data Used*

The data used in this analysis was generated by simulations of the *Swift* pipeline – as described in Section 2 – for different settings of the GRB redshift and luminosity distribution functions (Equations 2 and 3 in Lien et al. (2014) and reproduced below).

$$R_{\rm GRB}(z) = n_0 \begin{cases} (1+z)^{n_1} & z \le z_1 \\ (1+z_1)^{n_1-n_2}(1+z)^{n_2} & z > z_1 \end{cases} \tag{13}$$

$$\phi(L) = \frac{dN}{dL} = \begin{cases} \left(\frac{L}{L_\star}\right)^x & L \le L_\star \\ \left(\frac{L}{L_\star}\right)^y & L > L_\star \end{cases} \tag{14}$$

$R_{\rm GRB}(z)$ is the comoving GRB rate, with units of $\mathrm{Gpc^{-3}yr^{-1}}$. In these data sets, the luminosity distribution function was held constant with $x = -0.65$, $y = -3.00$, and $L_\star = 10^{52.05}$ erg/s. Additionally, the break in the redshift distribution was also held constant at $z_1 = 3.60$. Therefore, we only varied values of n_1 and n_2 (n_0 is ignored for the purpose of generating training data as it is only a normalization parameter). In total, 38 datasets are combined for use in training. These datasets were originally generated for Lien et al. (2014) and do not cover the space systematically. We use 34 of the 38 data sets for training models, including optimization of hyper-parameters; each of these contains ~ 4000 samples. The final 4, which contain ~ 10000 samples each, are set aside for evaluating the final model from each MLA as the validation data. The distribution of parameters for each of these data sets is shown in Figure 3.

We used this data for training as it was generated around the best-fit values from Lien et al. (2014) for the real *Swift* GRB redshift measurements of Fynbo et al. (2009). In the end, our goal is to fit the GRB rate model to these same observations.

A total of 15 parameters are taken from each simulated GRB in order to determine whether or not the GRB was detected by *Swift*. These are summarized in Table 1. These are used for classification of GRBs by MLAs. The target value is given by the trigger_index, which is 0 for GRBs that are not detected by the *Swift* algorithm and 1 for those that are detected.

A pair-wise plot of a few of the most significant parameters in determining detection is shown in Figure 4. Lighter points are GRBs that are detected by *Swift* in the trigger simulator (Lien et al. 2014) while darker ones are undetected GRBs. This plot shows a random subset of 5000 points from the entire training data set.

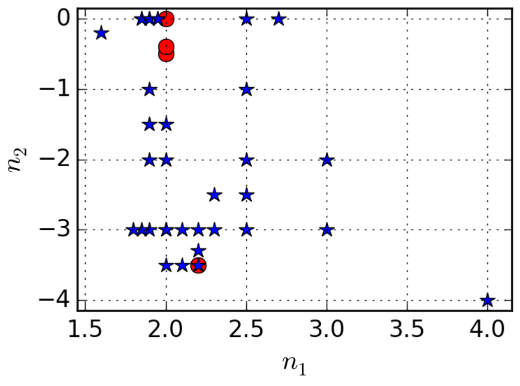

Figure 3. Values of the parameters for the redshift distribution function for sample GRB populations used to train ML models. Blue stars were used in training and optimization, red circles were used for final evaluation.

Parameter	Description
$\log_{10}(L)$	luminosity of the GRB
z	redshift
r	distance from center of detector grid of peak
ϕ	azimuthal angle in detector grid of peak
bin_size_emit	source time bin size
α	Band function parameter
β	Band function parameter
$\log_{10}(E_{\text{peak}})$	peak of the energy spectrum of the GRB
bgd_15-25keV	background count rate in 15–25keV band
bgd_15-50keV	background count rate in 15–50keV band
bgd_25-100keV	background count rate in 25–100keV band
bgd_50-350keV	background count rate in 50–350keV band
θ	incoming angle of GRB
$\log_{10}(\Phi)$	incident flux of GRB
ndet	number of active detector pixels (constant)
trigger_index	0 for non-detections and 1 for detections

Table 1
Parameters describing each simulated GRB. There are 15 inputs and the output class label. See Equation 4 in Lien et al. (2014) for details of α, β, and $\log(E_{\text{peak}})$.

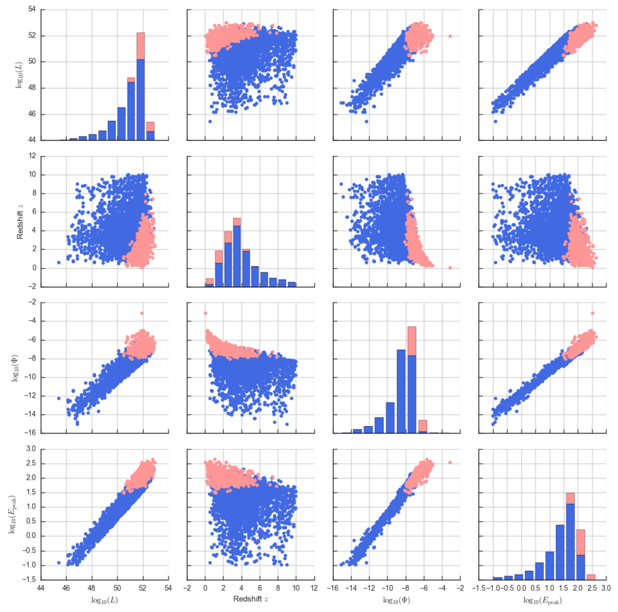

Figure 4. Pair-wise scatterplot of a few of the most significant parameters in determining detection. A random subset of 5000 points from the training data set is shown. GRBs that are detected are indicated by light red points, non-detected ones are blue.

To determine how much training data is required, we evaluated the learning curve for the random forest classifier. This plots the prediction accuracies, computed using 5-fold cross-validation, as a function of the size of the training data. The "training data" in this case is the $^4/_5$ of the data used for fitting the model and the "test data" is the $^1/_5$ left out for evaluation. The learning curve was done after finding the optimal RF settings in Section 4.2 as a check. We thus examined if use of the entire data set benefits model fitting significantly. The data was randomly shuffled before performing this test.

The resulting learning curve is shown in Figure 5. For small sample sizes, there is overfitting of the training data that begins to flatten out by 3×10^4 samples. The accuracy of the test set continues to increase as we add more data points, meaning that more data improves the generalizability of the model. Therefore, in all subsequent training we will use the entire data set for fitting a model; using fewer points would increase the bias of subsequent predictions.

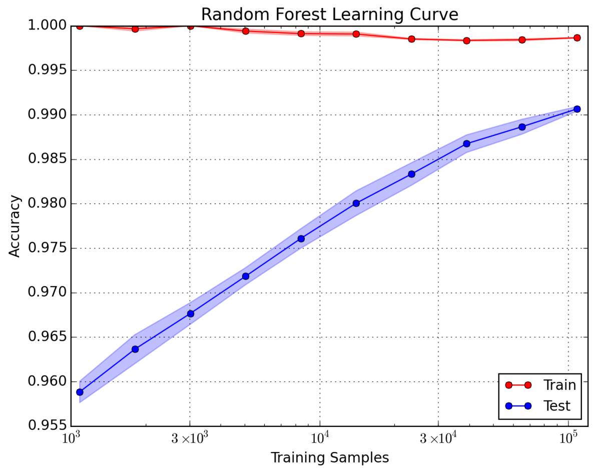

Figure 5. Learning curve for the random forest classifier. The training data set accuracy is fairly constant above 3×10^4 samples but the test accuracy continues to increase with the number of data points.

4.2. *Random Forest*

The random forest model was optimized for combinations of the min_samples_split and max_features parameters. These govern the minimum number of samples needed to perform a branching split and the number of features considered at each split, respectively. Choices for each are as follows[4]:

$$\text{min\_samples\_split} \in \{2, 4, 8, 16, 32, 64\}$$
$$\text{max\_features} \in \{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$$

Forests were trained with 500 trees, using the Gini impurity for deciding the optimal split at each branching point and with no limit on the number of branches before reaching a leaf. The 5-fold cross-validation evaluates the test accuracy for each pairwise combination of the parameters; the set with the highest test accuracy is the optimal model. The optimal parameters found were min_samples_split $= 4$ and

[4] The values for min_samples_split go from the absolute minimum, 2, to a significantly larger value where we see degraded performance by powers of 2. The choices for max_features vary from a low number (minimum is 1) to the maximum value that doesn't consider every parameter at each split and thus would have no randomness.

$\mathrm{max\_features} = 5$, however, it can be seen in Figure 6 that there is very little variation in accuracy with regard to the value of max_features. The minimum number of samples required to make a split is the dominant factor for improving the accuracy, where smaller values that naturally fine-tune the model further obtain better accuracy on the test set as well. The overall range in test set accuracy is not large and the worst model hyper-parameters still achieve accuracy $> 98\%$. The preference for lower values in max_features can be understood as increasing variability between trees in the forest and thus minimizing over-fitting.

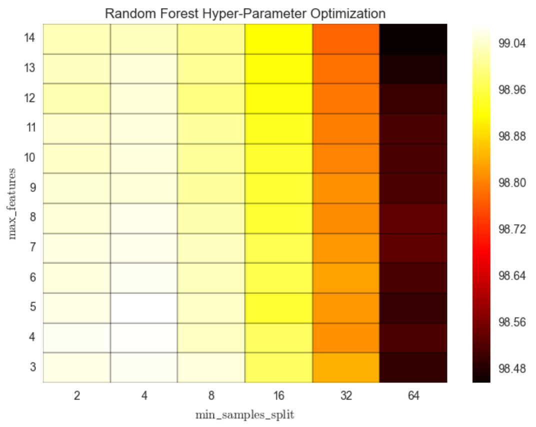

Figure 6. Test set accuracy for random forest classifier hyper-parameters. The optimal value is $(4, 5)$. It is clear that min_samples_split is a much stronger influence on over-fitting to the training data.

Using the optimal model, we perform predictions on the validation data set. With a naïve threshold of 0.5 on the output probability for the detection class for declaring a GRB detected, these predictions have an accuracy of 97.5%. This is lower than the test accuracy obtained earlier as the test samples were from the same distribution as the training ones while this validation data presents new distributions. The ROC for this classifier is shown with the others in Figure 9 and has an $\mathrm{AUC} = 0.9935$. Analysis of the F1-score found no significant difference between the optimal probability threshold and the naïve threshold of 0.5.

4.3. *AdaBoost*

The AdaBoost model was optimized for combinations of the n_estimators and learning_rate parameters. The former describes the number of 'weak learners' (decision trees) fit in each ensemble model and the latter describes the rate for adjusting the weighting of the weak learners as each is added to the ensemble. Settings for the individual decision trees were chosen to match those found as optimal for the random forest classifier, with $\mathrm{min\_samples\_split} = 4$ and $\mathrm{max\_features} = 5$. Choices for each were as follows[5]

$$\mathrm{n\_estimators} \in \{100, 200, 300, 400, 500\}$$
$$\log_{10}(\mathrm{learning\_rate}) \in \{-3, -2.5, -2, -1.5, -1, -0.5, 0\}$$

[5] The n_estimators range was determined by having enough trees for refined probability estimates while not needing more than the RF model. The range for the learning_rate parameter goes from a large value, 1, down to a small rate; we did not test smaller values as all models achieved very similar performance with each other and with the best RF model.

The 5-fold cross-validation found that the optimal parameters are $(100, 0.001)$. However, the range in test set accuracies is extremely small, varying only between 99.01% and 99.05%. Therefore, any of these models would be nearly equally accurate.

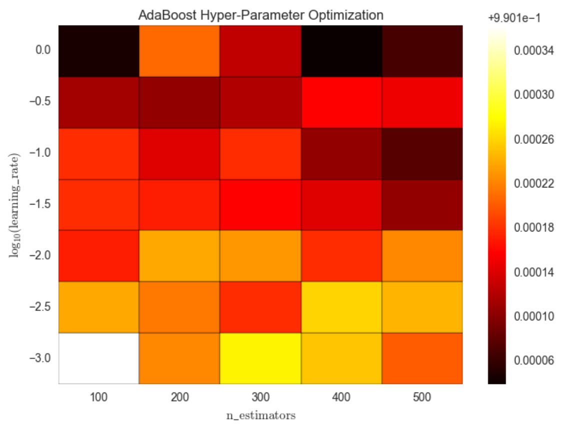

Figure 7. Test set accuracy for AdaBoost classifier hyper-parameters. The optimal value is $(100, 0.001)$. All options are very close to each other, ranging only from 99.01% to 99.05% in accuracy.

Using the optimal model, we perform predictions on the validation data set. With a naïve threshold of 0.5 on the output probability for the detection class for declaring a GRB detected, these predictions have an accuracy of 97.4%. The ROC for this classifier is shown with the others in Figure 9 and has an $\mathrm{AUC} = 0.9921$. Analysis of the F1-score found no significant difference between the optimal probability threshold and the naïve threshold of 0.5.

4.4. *Support Vector Machines*

The support vector machine model was trained using a Gaussian (radial basis function) kernel, as described in Equation (7). The input data values were all scaled to have zero mean and unit variance, so as to prevent undue bias in the kernel's distance measure. As errors in the predictions are allowed, there are thus two hyper-parameters to optimize, the penalty factor for errors, C, and the tunable parameter for the width of the Gaussian, γ. Choices examined for these were (after first searching over a larger grid with coarser spacing):

$$\log_{10}(C) \in \{1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3\}$$
$$\log_{10}(\gamma) \in \{-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1\}$$

5-fold cross validation found optimal parameters of $(C, \gamma) = (10^{2.25}, 1)$ with a test set accuracy of 99.0%. For smaller values of C, there is a much more limited range in γ that gives comparable results, if at all.

Using the optimal model and a 0.5 probability threshold for classification as a detection, the SVM has a prediction accuracy of 94.5%. The ROC for this classifier is shown in Figure 9 and has an $\mathrm{AUC} = 0.9348$. From all of these measures it is clear that the SVM model, in this scenario, does not generalize as well as the decision tree ensemble methods (RF and AdaBoost).

4.5. *Neural Networks*

Using SkyNet, we trained several neural network architectures using either the sigmoid or rectified linear activation

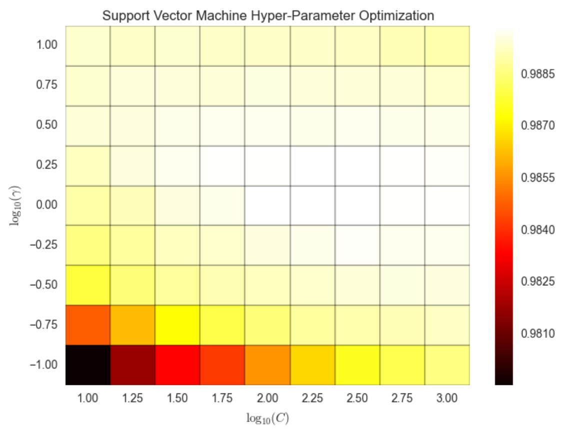

Figure 8. Test set accuracy for support vector machine classifier hyper-parameters. The optimal value is $(C, \gamma) = (10^{2.25}, 1)$.

Hidden Layers	Activation	Test Accuracy
25	sigmoid	97.89
	rectified	97.25
50	sigmoid	98.33
	rectified	97.57
100	sigmoid	98.47
	rectified	98.00
1000	sigmoid	97.49
	rectified	98.28
25+25	sigmoid	98.33
	rectified	97.65
50+50	sigmoid	98.73
	rectified	98.16
100+30	sigmoid	98.47
	rectified	98.27
100+50	sigmoid	98.64
	rectified	98.41
100+100	sigmoid	97.95
	rectified	98.35

Table 2
Test set accuracy from 5-fold cross-validation from the training of neural networks with SKYNET. The activation functions are given in Equation (10).

function for the hidden layer nodes. Training NNs is much more computationally expensive than any of the other models, despite the efficiencies in the training algorithm. Therefore, the size of our NN models (in both number and width of hidden layers) as well as the time spent training them, is limited. For each architecture[6] we employed 5-fold cross validation in order to asses its performance. We report in Table 2 the test set accuracies for each of the networks trained. They are all similar and are getting close to the 99% achieved by the previous MLAs. It is possible that more complex networks would achieve this level of accuracy.

Due to the constraints on training, we consider the NN architecture with highest average test accuracy, considering both activation functions: the 100+50 architecture of hidden layers. This is retrained on the entire data set with both activation functions and we find that the optimal model has hidden layers of 100+50 with the rectified linear unit activation function. We use this NN to make predictions on the validation data set with a naïve probability threshold of 0.5. This yields and accuracy of 96.9%. The ROC curve for this NN is shown in Figure 9 and has an AUC = 0.989. Analysis of the

[6] The architecture is given by X or X+Y, the former indicating a single hidden layer with X nodes and the latter indicating two hidden layers with X and Y nodes, respectively.

Classifier	Threshold	Accuracy	AUC	F1-score
Random Forest	0.449	0.975	0.994	0.912
AdaBoost	0.362	0.975	0.992	0.910
Neural Net	0.459	0.969	0.989	0.890
SVM	0.028	0.947	0.935	0.824
Flux	-7.243	0.896	0.945	0.663

Table 3
Results for measuring the performance of the classifiers trained in this study. The accuracy on the validation data, the area under the ROC curve, and the optimal F1-score are reported. The threshold values are probabilities for all models except the flux cut, which uses the $\log_{10}(\Phi)$.

F1-score found no significant difference between the optimal probability threshold and the naïve threshold of 0.5.

4.6. *Summary of Results*

Here we summarize the results for the optimal model returned by each MLA. The accuracy, AUC, and optimal F1-score are all reported in Table 3. We also include in this comparison the use of a constant cut in GRB flux; GRBs with flux greater than a threshold value will be labeled as detected and those with lower flux are non-detections. Varying this flux threshold produces a ROC and we find an optimal cut at $\log_{10}(\Phi) = -7.243$ erg/s/cm^2 (based on the F1-score) for which we measure the accuracy. It is clear that all ML classifiers except SVMs significantly outperform a flux threshold; SVMs still outperform a flux cut at optimal settings.

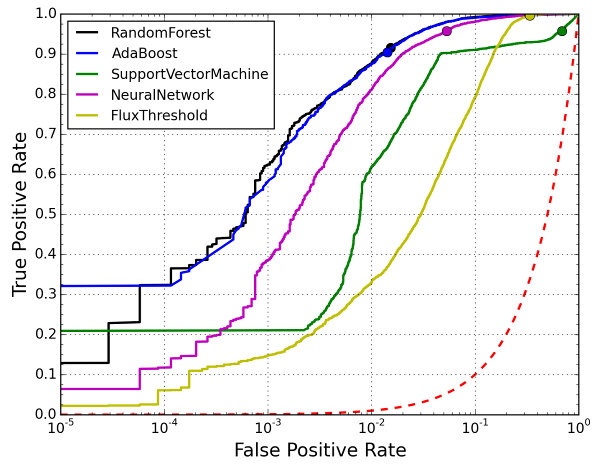

Figure 9. Receiver operating characteristic (ROC) curves for the classifiers. A dot is placed at the values for the optimal probability threshold found for each classifier. The ROC curve of a random classifier is shown in a dashed red line. A logarithmic scale for the x-axis is used to display the differences in the ROC curves.

From this analysis, we see that the RF and AdaBoost classifiers performed the best in classification task. NNs were very close behind, with SVMs performing the worst among the MLAs[7].

5. USE OF ACCELERATED PIPELINE FOR BAYESIAN INFERENCE

Here we demonstrate the use of the trained ML models in accelerating Bayesian inference, namely fitting the intrinsic redshift distribution of GRBs. We do so with best-fit random forest, AdaBoost, and SKYNET NN models, these being the most accurate.

[7] It should be noted that this is not a comment on the general performance of the MLAs, merely how well they performed on this task with this data set.

5.1. *Likelihood Function*

We first consider how we will evaluate the fit of a model to a set of GRB redshift observations, a.k.a. “the data”. If we bin the observations to obtain a redshift density, then in each bin (with central redshift z_i) there will be an observed number of GRBs, $N_{\rm obs}(z_i)$. There is also an expected number of intrinsic GRBs occurring in *Swift*'s field of view during the observation time in each redshift bin given by

$$N_{\rm int}(z_i) = \frac{4\pi}{6}\Delta t_{\rm obs} R_{{\rm GRB};dz}(z_i)dz, \tag{15}$$

where

$$R_{{\rm GRB};dz}(z) = \frac{R_{\rm GRB}(z)}{1+z}\frac{dV_{\rm comov}}{d\Omega dz}. \tag{16}$$

$R_{{\rm GRB};dz}(z)$ is the observed GRB rate that accounts for time dilation and the comoving volume in addition to the comoving rate, $R_{\rm GRB}(z)$. The $^{4\pi}/_{6}$ factor introduced here reflects that *Swift* observes only a sixth of the entire sky and $\Delta t_{\rm obs}$ reflects the fraction of time (per year) that *Swift* is observing; this is taken as $\Delta t_{\rm obs} \approx 0.8$ as calculated from related *Swift* log data. $V_{\rm comov}$ is the cosmological co-moving volume and Ω is the subtended sky angle.

Not all GRBs occurring in *Swift*'s field of view will be detected, however; this is taken into account by the extra factor, $F_{\rm det}(z)$. This is the fraction of GRBs at redshift z that are detected by *Swift* and is further discussed in Section 5.1.1. Including this factor gives us the expected number of observed GRBs in each bin,

$$N_{\rm exp}(z_i) = \frac{4\pi}{6}\Delta t_{\rm obs} R_{{\rm GRB};dz}(z_i)F_{\rm det}(z_i)dz. \tag{17}$$

The probability, then, of observing $N_{\rm obs}(z_i)$ GRBs when $N_{\rm exp}(z_i)$ are expected is given by the Poisson distribution. The bins can be treated as independent, so for K bins we can multiply their probabilities.

$$\begin{aligned}\Pr(\{N_{\rm obs}(z_i)\};\{N_{\rm exp}(z_i)\}) &= \prod_{i=1}^{K}\Pr(N_{\rm obs}(z_i);N_{\rm exp}(z_i)) \\ &= \prod_{i=1}^{K}\frac{N_{\rm exp}(z_i)^{N_{\rm obs}(z_i)}e^{-N_{\rm exp}(z_i)}}{N_{\rm obs}(z_i)!}\end{aligned} \tag{18}$$

The log-likelihood is therefore the log of this probability,

$$\begin{aligned}\mathcal{L}(\vec{n}) &= \log\left(\Pr(N_{\rm obs}(z_i);N_{\rm exp}(z_i))\right) \\ &= \sum_{i=1}^{K} N_{\rm obs}(z_i)\log(N_{\rm exp}(z_i)) - N_{\rm exp}(z_i) - \log(N_{\rm obs}(z_i)!)\end{aligned} \tag{19}$$

where $\vec{n} = \{n_0, n_1, n_2, z_1, x, y, L_*\}$ is the set of model parameters that let us obtain $N_{\rm exp}(z_i)$, which is really $N_{\rm exp}(z_i|\vec{n})$.

In the limit of a large number of bins, each bin will contain either 0 or 1 detected GRBs so $N_{\rm obs}(z_i)! = 1 \Rightarrow \log(N_{\rm obs}(z_i)!) = 0$. We can also split terms and rewrite Eq (19) as

$$\begin{aligned}\mathcal{L}(\vec{n}) &= \sum_{i=1}^{K}\left[N_{\rm obs}(z_i)\log(N_{\rm exp}(z_i))\right] - \sum_{i=1}^{K}N_{\rm exp}(z_i) \\ &= -N_{\rm exp} + \sum_{\{i\}_{\rm det}}\log(N_{\rm exp}(z_i))\end{aligned} \tag{20}$$

where $\{i\}_{\rm det}$ are those bins with a detection. We can perform this calculation in the limit of infinite bins, essentially a continuous measurement. $N_{\rm exp}$ is the integrated expected rate of observations given by

$$N_{\rm exp} = \int_0^{10} N_{\rm exp}(z)dz. \tag{21}$$

This likelihood is the same as the C-statistic derived in Cash (1979) in the un-binned limit (see Equation 7 therein, where $C = -2\mathcal{L}$). This likelihood function is also equivalent to that of Stevenson et al. (2015), which compares discrete intrinsic population models for binary black hole mergers as observed by advanced LIGO and Virgo using the observed mass distribution, if the latter is taken to the same limit of infinite bins of infinitesimal width. This is particularly notable as Stevenson et al. (2015) uses a Poisson probability for the total number of detections multiplied by a multinomial distribution describing the fractional distribution of detections among bins in mass space.

5.1.1. *Detection Fraction*

The detection fraction (also known as detection efficiency) $F_{\rm det}(z)$ is computed in advance of the analysis by utilizing the ML models trained to reproduce the *Swift* detection pipeline. 10^6 GRBs are simulated at each of 10,001 redshift points in $[0, 10]$ in order to precisely measure the average detection fraction. These points are used as the basis for a spline interpolation to compute $F_{\rm det}(z)$ at any z. The detection fraction as a function of z from each the three models used is shown in Figure 10. It is important to note that this $F_{\rm det}(z)$ is calculated under the assumption of the particular luminosity function used in this study; it may change significantly for other choices of the luminosity function parameters.

We also show, for comparison, the detection fraction as computed by the constant flux cut and from an analytic fit used in Howell et al. (2014). This was computed using the data from Lien et al. (2014), so we are not surprised that it matches well in the low-redshift range where there is better sampling. The flux cut has discrepancies across the entire redshift range while the analytic fit is close until $z = 5.96$, after which the authors used a constant value.

These can all be compared against the detection fraction of the entire data set (training and validation) provided from using the original *Swift* pipeline of Lien et al. (2014). There is less resolution and large uncertainty on this curve as there are much fewer samples ($\mathcal{O}(10^5)$ vs $\mathcal{O}(10^9)$), but we can see that RF, AB, and NN track it well.

5.2. *Model, Parameters, and Prior*

In our analysis, as the detection fraction is averaged over the luminosity distribution, we hold those parameters constant with $x = -0.65$, $y = -3.00$, and $L_\star = 10^{52.05}$ erg/s. The parameters describing the redshift distribution are allowed to vary with ranges and prior distribution given in Table 4.

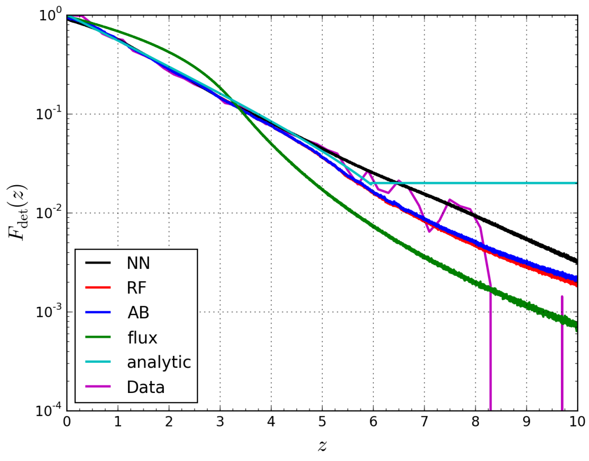

Figure 10. $F_{\rm det}(z)$ as computed by the three different MLAs used as well as the constant flux cut and an analytic form used in Howell et al. (2014). The detection fraction of all data provided for training and validation is also shown. This is calculated under the assumption of the particular luminosity function used in this study and may change significantly for other choices of the luminosity function parameters.

Parameter	Min	Max	Prior
n_0	0.01	2.00	logarithmic
n_1	0.00	4.00	flat
n_2	-6.00	0.00	flat
z_1	0.00	10.00	flat

Table 4
Prior ranges and distributions for the redshift distribution model parameters.

The population generation code developed in Lien et al. (2014) was used to generate simulated data for testing purposes. In addition to the above-specified parameters, we also return the total number of GRBs, $N_{\rm exp}$.

5.3. *Parameter Estimation Tests*

The BAMBI algorithm (Feroz and Hobson 2008; Feroz et al. 2009; Graff et al. 2012) is a general-purpose implementation of the nested sampling algorithm for Bayesian inference. We use it to perform Bayesian parameter estimation, measuring the full posterior probability distribution of the model parameters.

In the ideal case, any $X\%$ credible interval calculated from the posterior distribution should contain the true parameters $\sim X\%$ of the time. We sampled a large number of parameter values from the prior and obtained a posterior distribution from simulated data generated with each. For each parameter, we then computed the cumulative fraction of times the true value was found at a credible interval of p - as integrated up from the minimum value - as a function of p. This result was compared to a perfect one-to-one relation using the Kolmogorov-Smirnov test. All parameters passed this test, thus confirming the validity of returned credible intervals.

The posterior distribution for a particular realization of an observed GRB redshift distribution generated using $\{n_0, n_1, n_2, z_1\} = \{0.42, 2.07, -0.70, 3.60\}$ (best-fit values from Lien et al. (2014)) is shown in Figures 11, 12, and 13 for the random forest, AdaBoost, and SkyNet NN models, respectively. While the random forest and AdaBoost posteriors are nearly identical, the SkyNet posterior has small differences due to the difference in detection fraction. However, these differences are not major. We can see that n_2 is effectively unconstrained due to the low number of observed GRBs with redshift greater than z_1. The true values are marked by

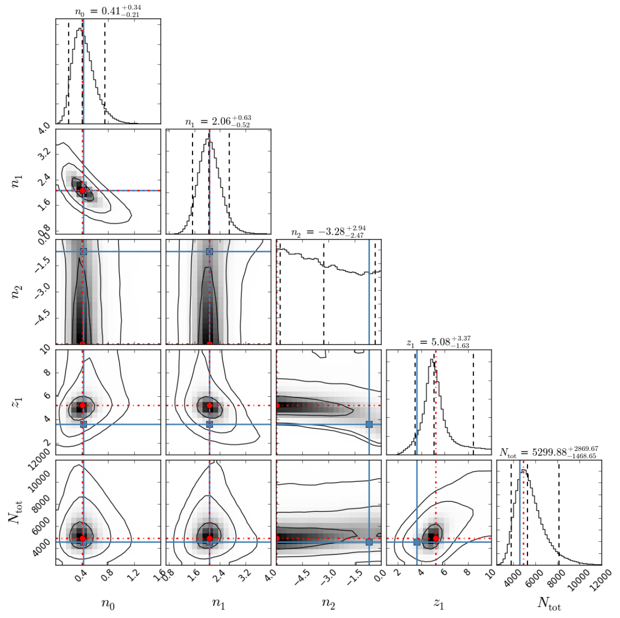

Figure 11. Posterior distribution for simulated data with $\{n_0, n_1, n_2, z_1\} = \{0.42, 2.07, -0.70, 3.60\}$ using the random forest classifier for data generation and detection fraction. $N_{\rm tot}$ is the total number of GRBs in the Universe per year. Blue lines indicate true values and dot-dash red lines indicate maximum likelihood (i.e. best-fit) values. 2D plots show contour lines every σ (68%, 95%, 99%). Vertical dashed lines in 1D plots show 5%, 50%, and 95% quantiles, with values given in the titles.

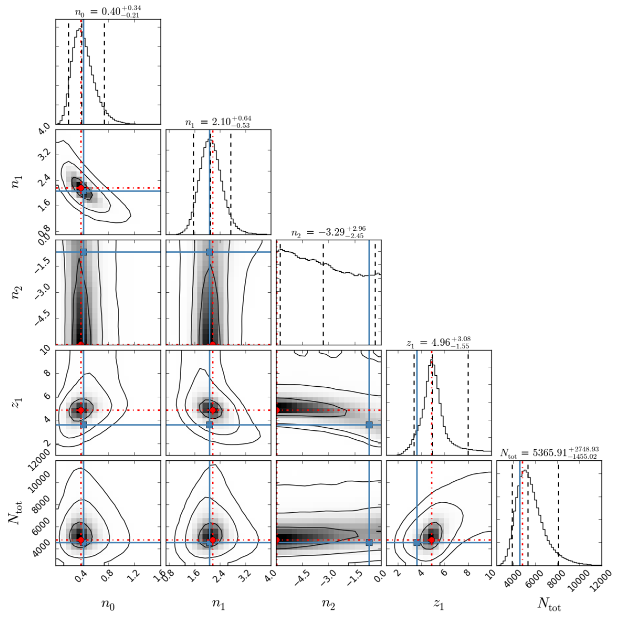

Figure 12. Posterior distribution for simulated data with $\{n_0, n_1, n_2, z_1\} = \{0.42, 2.07, -0.70, 3.60\}$ using the AdaBoost classifier for data generation and detection fraction. Same features as Figure 11.

the blue lines.

We also plot in Figure 14 the distribution of model predictions as specified by the posterior (from RF). In both panels, we select 200 random models selected from the set of posterior samples (light blue lines) as well as the maximum $\mathcal{L}(\vec{n})$ point (black line). The upper panel shows $R_{\rm GRB}(z)$ (Equation (13)); the lower panel shows $N_{\rm exp}(z)/dz$ (Equation (17)) and $N_{\rm int}(z)/dz$. The lower panel also plots a histogram of the simulated population of measured redshifts for observed

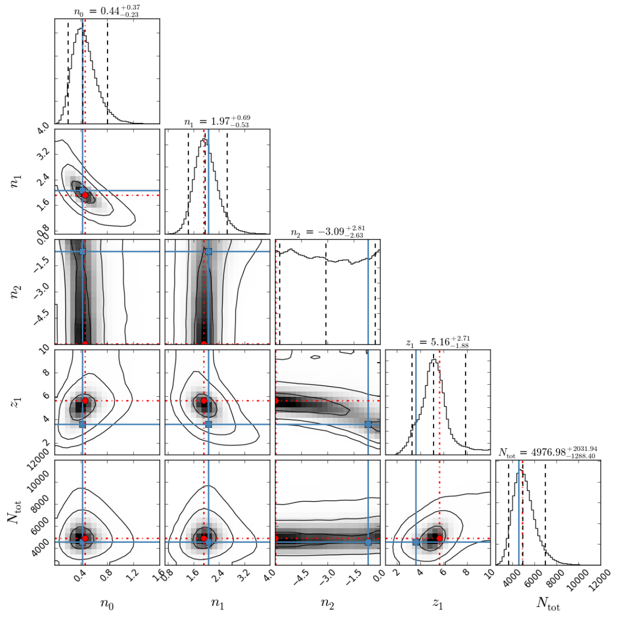

Figure 13. Posterior distribution for simulated data with $\{n_0, n_1, n_2, z_1\} = \{0.42, 2.07, -0.70, 3.60\}$ using the SkyNet NN classifier for data generation and detection fraction. Same features as Figure 11.

GRBs. The upper panel clearly shows us the allowed variability in the high redshift predictions of the model; in the lower panel, we see that the detection fraction and other factors constrain this variability to consistently low predictions.

These tests show that we can trust the results of an analysis – under the model assumptions, we can recover the true parameters of a simulated GRB redshift distribution.

5.4. *Analysis of Swift GRBs*

In Lien et al. (2014), the authors use a sample of 66 GRBs observed by *Swift* whose redshift has been measured from afterglows only or afterglows and host galaxy observations. These observations are taken from the larger set of Fynbo et al. (2009) and the selection is done in order to remove bias towards lower-redshift GRBs in the fraction with measured redshifts (see Section 4.1 of Lien et al. (2014)). In our final analysis, we use these 66 GRB redshift measurements as data that we fit with the models described in this paper.

Using random forests, AdaBoost, and neural network ML models for the detection fraction, we find posterior probability distributions for n_0, n_1, n_2, and z_1, as seen in Figures 15, 16, and 17, respectively. The maximum likelihood estimate and posterior probability central 90% credible interval are given in Table 5. We also plot in Figure 18 the distribution of model predictions as specified by the posterior (from RF) as we did in Figure 14 for the test population.

Parameters n_0, n_1, and $N_{\rm tot}$ show mostly Gaussian marginal distributions and some correlation between n_0 and n_1 – larger values of the former lead to lower values of the latter in order to maintain a constant value for $N_{\rm tot}$ and similar values at the peak of the observed distribution. The data do not strongly constrain the high redshift part of the distribution, namely the n_2 parameter. The upper panel of Figure 18 clearly shows us the allowed variability in the high redshift predictions of the model; in the lower panel, we see that the detection fraction and other factors constrain this variability to consistently low predicted numbers of GRB observations. We see a double-peak in z_1, not the clear single peak seen in the simulated data. One peak occurs around $z_1 \approx 3.6$, the best-fit value from Lien et al. (2014) and is more prominent when using the NN model. This shows a sensitivity to the detection fraction for this set of GRB observations. A hint of this can be seen in the posterior plots of Section 5.3 – Figures 11, 12, and 13. All measured parameters are consistent with the best-fit values found by Lien et al. (2014).

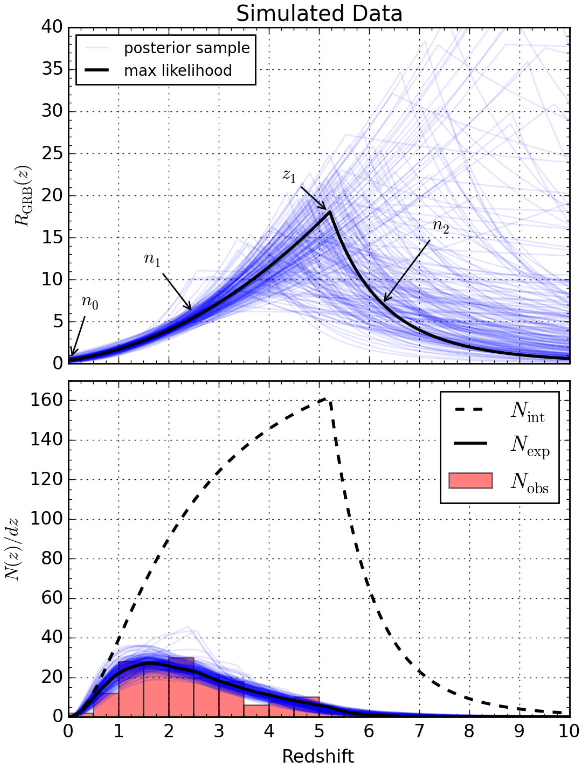

Figure 14. The distribution of model predictions from the posterior (RF) for a simulated population of GRBs. 200 models with parameters chosen randomly from the posterior are shown in light blue lines in both panels. The maximum $\mathcal{L}(\vec{n})$ point is shown in black. The upper panel shows $R_{\rm GRB}(z)$ (Equation (13)) and the lower panel shows $N_{\rm exp}(z)/dz$ (Equation (17)). The lower panel also shows the simulated population of measured redshifts for observed GRBs and $N_{\rm int}(z)/dz$ for the maximum $\mathcal{L}(\vec{n})$ point in dashed black.

Parameter	Method	Max Like	90% CI
n_0	RF	0.480	[0.247, 0.890]
	AB	0.489	[0.249, 0.902]
	NN	0.416	[0.238, 0.986]
n_1	RF	1.700	[1.155, 2.261]
	AB	1.681	[1.146, 2.273]
	NN	1.875	[1.030, 2.334]
n_2	RF	-5.934	[-5.675, -0.238]
	AB	-5.950	[-5.665, -0.230]
	NN	-0.483	[-5.598, -0.217]
z_1	RF	6.857	[3.682, 9.654]
	AB	6.682	[3.603, 9.622]
	NN	3.418	[3.215, 9.385]
$N_{\rm exp}$	RF	4455	[2967, 6942]
	AB	4392	[2967, 6822]
	NN	3421	[2546, 5502]

Table 5
Maximum likelihood (i.e. best-fit) estimates and central 90% credible intervals for the redshift distribution parameters as fit to the real set of 66 *Swift* GRBs (Fynbo et al. 2009; Lien et al. 2014) using each of the MLAs.

5.5. *Computational Cost*

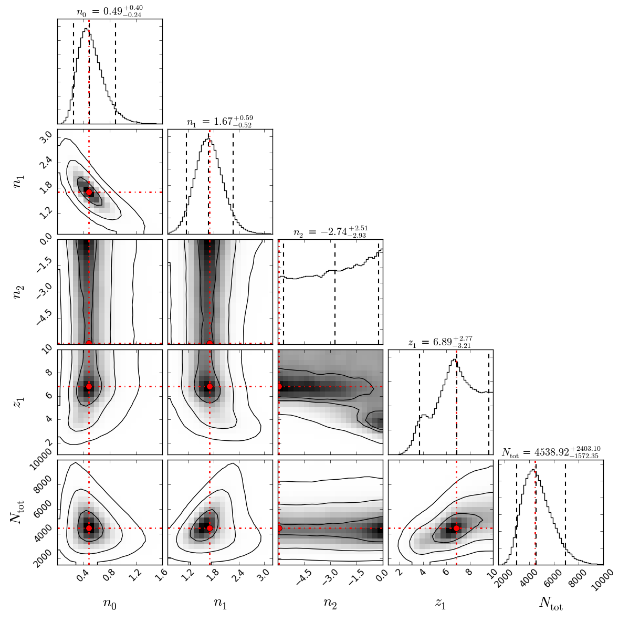

Figure 15. Posterior distribution for the real set of 66 *Swift* GRBs using the random forest classifier for the detection fraction. $N_{\rm tot}$ is the total number of GRBs in the Universe per year. The dot-dash red lines indicate maximum likelihood (i.e. best-fit) values. 2D plots show contour lines every σ (68%, 95%, 99%). Vertical dashed lines in 1D plots show 5%, 50%, and 95% quantiles, with values given in the titles.

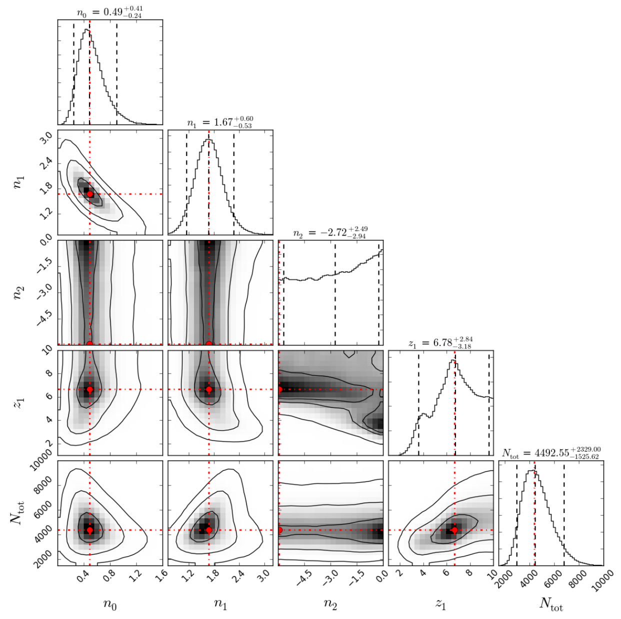

Figure 16. Posterior distribution for the real set of 66 *Swift* GRBs using the AdaBoost classifier for the detection fraction. Similar to Figure 15.

The main computational costs of this entire analysis procedure were:

1. Producing the training data
2. Performing MLA model fitting and hyper-parameter optimization
3. Using the MLA models to compute the detection fraction.

These steps are in roughly decreasing order of cost, from CPU weeks to days. However, all three are one-time initialization

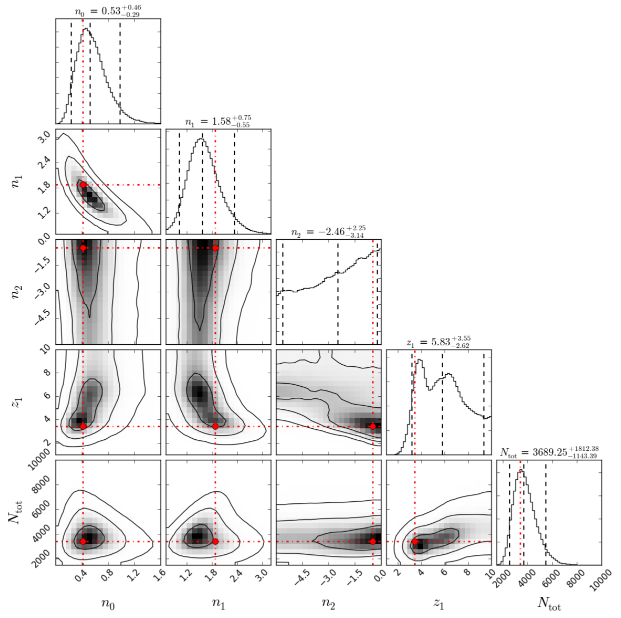

Figure 17. Posterior distribution for the real set of 66 *Swift* GRBs using the SkyNet NN classifier for the detection fraction. Similar to Figure 15.

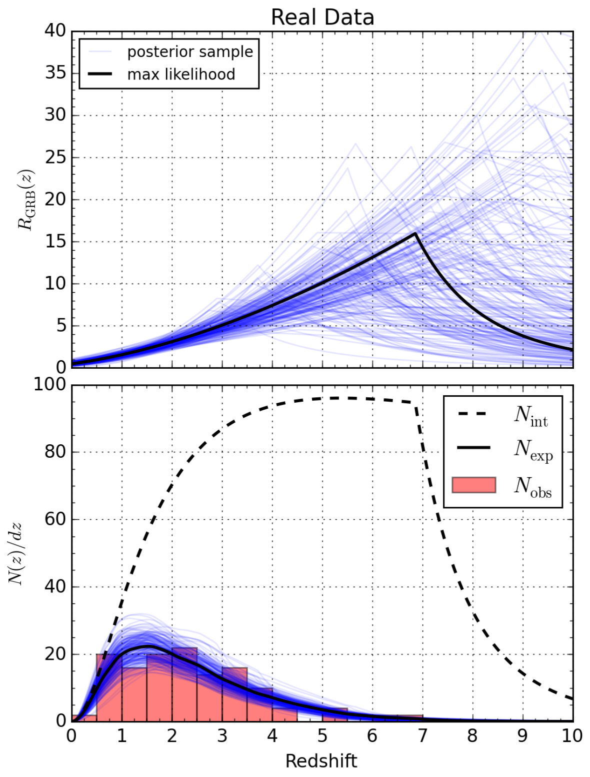

Figure 18. The distribution of model predictions from the posterior (RF) for the real set of 66 *Swift* GRBs (Fynbo et al. 2009). 200 models with parameters chosen randomly from the posterior are shown in light blue lines in both panels. The maximum $\mathcal{L}(\vec{n})$ point is shown in black. The upper panel shows $R_{\rm GRB}(z)$ (Equation (13)) and the lower panel shows $N_{\rm exp}(z)/dz$ (Equation (17)). The lower panel also shows the distribution of measured redshifts for observed GRBs and $N_{\rm int}(z)/dz$ for the maximum $\mathcal{L}(\vec{n})$ point in dashed black.

costs and can be run massively parallel to reduce wall-time.

After this initialization is complete, however, subsequent analysis of real or simulated data is performed extremely quickly. A single likelihood evaluation takes < 0.1 ms, meaning that a Bayesian analysis can be computed in less than a minute on a laptop. Providing this same kind of accurate measurement of the detection fraction without the MLAs would take orders of magnitude more time; while $\mathcal{O}(10^5)$ samples were used for training the MLA mdoels, $\mathcal{O}(10^{10})$ evaluations were used in measuring the detection fraction as a function of redshift. The precision of the detection fraction would need to be reduced significantly to make the overall cost comparable. Furthermore, we now are equipped with accurate models of the *Swift* detection algorithm.

6. COMPARISON TO PREVIOUS WORK

We have developed a machine learning algorithm (simulator) for the detailed *Swift* BAT long GRB pipeline simulator developed in Lien et al. (2014). These techniques allow us to complete a thorough Bayesian analysis of the long GRB rate redshift dependence using the Fynbo et al. (2009) data set, improving on the more coarsely sampled study in Lien et al. (2014). Our results are compatible with those from Lien et al. (2014) with tight agreement for lower redshifts up to $z \sim 4$ with compatible results and relatively narrow distributions for our n_1 and n_0 rate parameters. We find values of $n_0 \sim 0.48^{+0.41}_{-0.23}$ $\mathrm{Gpc}^{-3}\mathrm{yr}^{-1}$ and $n_1 \sim 1.7^{+0.6}_{-0.5}$, consistent with the best-fit values of $n_0 = 0.42$ and $n_1 = 2.07$ from Lien et al. (2014). For larger redshifts the model is less constrained; n_2 spans the prior range and z_1 is significantly constrained only at the low-z end. Our general agreement with Lien et al. (2014) supports their identification of differences between the long GRB redshift distribution and estimates of the star formation rate (Hopkins and Beacom 2006). Though our analysis indicates that the Fynbo et al. (2009) data do not provide strong constraints on the rate at high redshift the results seem to indicate significant differences for $z < 4$. A follow-up Bayesian analysis comparing with a two-break model would allow a more direct comparison with SFR models. We can also note how our results compare with several other studies which use GRB observations and subsequent redshift measurements in order to estimate the redshift or luminosity distribution of GRBs in the Universe.

The paper by Butler et al. (2010) used an extensive set of GRBs both with and without redshift measurements to fit intrinsic distributions for GRB redshift, luminosity, peak flux, and more. This fitting was performed using PyMC, a python package for Markov chain Monte Carlo analyses, marginalizing over all redshifts when no measurement is available; the log-likelihood function used is un-binned, similar to the one used in our study. The detection fraction (a.k.a. detection efficiency) used by Butler et al. (2010), however, is a probability dependent solely on the photon count rate. Their results for n_1, n_2, and z_1 are consistent with 90% confidence intervals that we measure.

Wanderman and Piran (2010) performs a careful study of the GRB rate and luminosity distribution via a Monte-Carlo approach. This study adopts an empirical probability function to determine whether a burst is detectable based on the peak flux. In addition, they also introduce an empirical function to estimate the probability of obtaining a redshift measurement based on the GRB peak flux. Since we adopt the same functional form as Wanderman and Piran (2010), it is possible to compare the values of the same parameters. However, in this paper we quantify the parameter uncertainties of the GRB rate, and assume an un-changed luminosity function from Lien et al. (2014), which is different the one found in Wanderman and Piran (2010). The parameters found by Wanderman and Piran (2010) are $n0 \sim 1.25$, $n1 \sim 2.07$, and $n_2 \sim -1.36$ (as listed in Table 2 of by Wanderman and Piran (2010)). These values of n_1 and n_2 are consistent with our findings; the value of n_0 is at the upper end of our range, but this difference is likely due to the difference in luminosity distribution.

Salvaterra et al. (2012) constructs a sub-sample of *Swift* long GRBs that is complete in redshift by selecting bursts that satisfy certain observational criteria that are optimal for follow-up observations. In addition, these authors select only bright bursts with 1-s peak photon fluxes greater than 2.6 photons s^{-1} cm^{-2}, in order to achieve a high completeness of 90% in redshift measurements. They use this sub-sample to estimate the luminosity function and GRB rate via maximum likelihood estimation – using the same likelihood as our study and marginalizing over a flat z distribution if no value was measured for a GRB –, and found that either the rate or the luminosity function is required to evolve strongly with redshift, in order to explain the observational data. The *Swift* detection efficiency is modeled as a threshold on the GRB flux. The rate model fits of Salvaterra et al. (2012) are not directly comparable to ours due to a different functional form based off of the SFR.

The study of Howell et al. (2014) takes advantage of some of the work done by Lien et al. (2014) in using the detection efficiency computed from simulated GRB populations. The authors perform a time-dependent analysis that considers the rarest events – the largest redshift or the highest peak flux – and how these values progress over observational time. These are used to fit the intrinsic redshift and luminosity distributions of GRBs and infer 90% confidence intervals. Howell et al. (2014) measures a local GRB rate density consistent with our constraints on n_0. Other rate parameters were held fixed to values obtained by Lien et al. (2014) and are thus also consistent with our measurements.

Yu et al. (2015) and Petrosian et al. (2015) use sub-sets of observed GRBs at different redshifts to construct a more complete GRB sample and account for observational biases. This method is called Lynden-Bells c^{-} method. Each sub-sample is selected based on the minimum detectable GRB luminosity at each redshift. Both of these studies find significant luminosity evolution and a rather high GRB rate at low redshift in comparison to the one expected from previous star-formation rate measurements. However, as noted in Yu et al. (2015) and Petrosian et al. (2015), several additional selection effects can be the cause of this discrepancy, including the potential bias toward redshift measurements of nearby GRBs and those with bright X-ray and optical afterglows. The rate evolution found by Yu et al. (2015) is not consistent with our results at low redshift, but is consistent at high redshift due to the large uncertainty in measuring n_2.

Our study is able to improve upon the methodology of these studies and may be extended to cover the same breadth of GRB source models to be fit. These improvements are not the same for all, but in summary involve using a fully Bayesian model fitting procedure with a likelihood function that does not involve any binning of observations. Furthermore, the detection efficiency of the *Swift* BAT detector can be better modeled using ML techniques that incorporate all available information (marginalizing over parameters not under consideration) than with probabilities dependent solely on the flux or photon counts. With both of these, not only will we be able to extract as much information as possible out of GRB detections and follow-up observations, but such analyses will incur minimal modeling bias while maintaining computational speed.

7. SUMMARY AND CONCLUSIONS

We have built a set of models emulating the *Swift* BAT detection algorithm for long GRBs using machine learning. Using a large set of simulated GRBs from the work of Lien et al. (2014) as training data, we used the random forest, AdaBoost, support vector machine, and neural network algorithms to optimize, fit, and validate models that simulate the *Swift* triggering algorithm to high accuracy. RF and AdaBoost perform best, achieving accuracies of 97.5%; NNs and SVMs have accuracies of 96.9% and 94.7%, respectively. These all outperform a threshold in GRB flux, which has an accuracy of 89.6%. The improved faithfulness to the full *Swift* triggering removes potential sources of bias when performing analyses based on the model.

Using these models, we computed the detection fraction (efficiency) of *Swift* as a function of redshift for a fixed luminosity distribution. Using this empirical detection fraction and a model for the GRB rate given by Equation (13), we fit the model parameters on both simulated redshift measurements and on the redshifts reported by Fynbo et al. (2009). We find best-fitting values and 90% credible intervals as reported in Table 5 for each of the top three MLAs. These, expectedly, are consistent with values found by Lien et al. (2014).

After incurring the initial costs of generating training data, fitting the models, and computing the detection fraction, we are able to perform Bayesian parameter estimation extremely rapidly. This allows us to explore the full parameter space of the model and determine not only the best-fit parameters but also the uncertainty and degeneracies present.

8. FUTURE WORK

In performing this analysis, we identified several potential avenues for further work to improve our model and extend the analysis performed. Hence, we see this work as just the first step in advancing GRB research. The easiest extension is to analyze different samples of measured GRB redshifts that may contain different selection biases.

To improve our ML models, we would like to continue building on our training data set and making it more agnostic with respect to GRB parameters. This would allow for better modeling of the *Swift* detection algorithm and its dependency on different GRB characteristics. The GRB rate model can also be expanded to include a second break point in redshift. This would allow for more direct comparison with most fits of the SFR that use a double-broken power-law model. Bayesian model selection could then be used to compare these and other models.

The analyses can be extended to fitting the intrinsic luminosity distribution by including GRB luminosity or flux in the detection fraction, $F_{\rm det}\left(\log_{10}(L), z\right)$ or $F_{\rm det}\left(\Phi\left(\log_{10}(L), z\right), z\right)$. The likelihood function can then jointly describe both the luminosity and redshift distributions – including luminosity distribution evolution with redshift – by analyzing measured GRB fluxes and redshifts; the redshift distribution can be marginalized over if there is no measured value for a particular GRB. The likelihood function can also be modified to account for known selection biases, including the probability of measuring a redshift for each GRB.

Beyond improving and extending the model used in this paper, a similar analysis can be performed for the study of short GRBs detected by *Swift* and other detectors. This work has demonstrated the value of machine learning for GRB data analysis and the algorithms and techniques may be extended to other problems in GRB follow-up and analysis.

ACKNOWLEDGEMENTS

The authors would like to thank Brad Cenko, Judith Racusin, and Neil Gehrels for helpful discussions. PG acknowledges support from NASA Grant NNX12AN10G and an appointment to the NASA Postdoctoral Program at the Goddard Space Flight Center, administered by Oak Ridge Associated Universities through a contract with NASA. JB acknowledges support from NASA Grant ATP11-00046.

REFERENCES

S. D. Barthelmy, L. M. Barbier, J. R. Cummings, E. E. Fenimore, N. Gehrels, D. Hullinger, H. A. Krimm, C. B. Markwardt, D. M. Palmer, A. Parsons, G. Sato, M. Suzuki, T. Takahashi, M. Tashiro, and J. Tueller. The Burst Alert Telescope (BAT) on the SWIFT Midex Mission. Space Sci. Rev., 120:143–164, October 2005. doi:10.1007/s11214-005-5096-3.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi:10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A%3A1010933404324`.

N. R. Butler, J. S. Bloom, and D. Poznanski. The Cosmic Rate, Luminosity Function, and Intrinsic Correlations of Long Gamma-Ray Bursts. ApJ, 711:495–516, March 2010. doi:10.1088/0004-637X/711/1/495.

M. A. Campisi, L.-X. Li, and P. Jakobsson. Redshift distribution and luminosity function of long gamma-ray bursts from cosmological simulations. MNRAS, 407:1972–1980, September 2010. doi:10.1111/j.1365-2966.2010.17044.x.

W. Cash. Parameter estimation in astronomy through application of the likelihood ratio. ApJ, 228:939–947, March 1979. doi:10.1086/156922.

Wikimedia Commons. Svm max sep hyperplane with margin, 2008. URL `https://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ISSN 0885-6125. doi:10.1007/BF00994018. URL `http://dx.doi.org/10.1007/BF00994018`.

D. M. Coward, E. J. Howell, M. Branchesi, G. Stratta, D. Guetta, B. Gendre, and D. Macpherson. The Swift gamma-ray burst redshift distribution: selection biases and optical brightness evolution at high z? MNRAS, 432: 2141–2149, July 2013. doi:10.1093/mnras/stt537.

E. E. Fenimore, D. Palmer, M. Galassi, T. Tavenner, S. Barthelmy, N. Gehrels, A. Parsons, and J. Tueller. The Trigger Algorithm for the Burst Alert Telescope on Swift. In G. R. Ricker and R. K. Vanderspek, editors, *Gamma-Ray Burst and Afterglow Astronomy 2001: A Workshop Celebrating the First Year of the HETE Mission*, volume 662 of *American Institute of Physics Conference Series*, pages 491–493, April 2003. doi:10.1063/1.1579409.

E. E. Fenimore, K. McLean, D. Palmer, S. Barthelmy, N. Gehrels, H. Krimm, C. Markwardt, A. Parsons, and J. Tueller. Swift's Ability to Detect Gamma-Ray Bursts. *Baltic Astronomy*, 13:301–306, 2004.

F. Feroz and M. P. Hobson. Multimodal nested sampling: an efficient and robust alternative to Markov Chain Monte Carlo methods for astronomical data analyses. *MNRAS*, 384:449–463, February 2008. doi:10.1111/j.1365-2966.2007.12353.x.

F. Feroz, M. P. Hobson, and M. Bridges. MULTINEST: an efficient and robust Bayesian inference tool for cosmology and particle physics. *MNRAS*, 398:1601–1614, October 2009. doi:10.1111/j.1365-2966.2009.14548.x.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997. ISSN 0022-0000. doi:http://dx.doi.org/10.1006/jcss.1997.1504. URL `http://www.sciencedirect.com/science/article/pii/S002200009791504X`.

J. P. U. Fynbo, P. Jakobsson, J. X. Prochaska, D. Malesani, C. Ledoux, A. de Ugarte Postigo, M. Nardini, P. M. Vreeswijk, K. Wiersema, J. Hjorth, J. Sollerman, H.-W. Chen, C. C. Thöne, G. Björnsson, J. S. Bloom, A. J. Castro-Tirado, L. Christensen, A. De Cia, A. S. Fruchter, J. Gorosabel, J. F. Graham, A. O. Jaunsen, B. L. Jensen, D. A. Kann, C. Kouveliotou, A. J. Levan, J. Maund, N. Masetti, B. Milvang-Jensen, E. Palazzi, D. A. Perley, E. Pian, E. Rol, P. Schady, R. L. C. Starling, N. R. Tanvir, D. J. Watson, D. Xu, T. Augusteijn, F. Grundahl, J. Telting, and P.-O. Quirion. Low-resolution Spectroscopy of Gamma-ray Burst Optical Afterglows: Biases in the Swift Sample and Characterization of the Absorbers. ApJS, 185:526–573, December 2009. doi:10.1088/0067-0049/185/2/526.

N. Gehrels, G. Chincarini, P. Giommi, K. O. Mason, J. A. Nousek, A. A. Wells, N. E. White, S. D. Barthelmy, D. N. Burrows, L. R. Cominsky, K. C. Hurley, F. E. Marshall, P. Mészáros, P. W. A. Roming, L. Angelini, L. M. Barbier, T. Belloni, S. Campana, P. A. Caraveo, M. M. Chester, O. Citterio, T. L. Cline, M. S. Cropper, J. R. Cummings, A. J. Dean, E. D. Feigelson, E. E. Fenimore, D. A. Frail, A. S. Fruchter, G. P. Garmire, K. Gendreau, G. Ghisellini, J. Greiner, J. E. Hill, S. D. Hunsberger, H. A. Krimm, S. R. Kulkarni, P. Kumar, F. Lebrun, N. M. Lloyd-Ronning, C. B. Markwardt, B. J. Mattson, R. F. Mushotzky, J. P. Norris, J. Osborne, B. Paczynski, D. M. Palmer, H.-S. Park, A. M. Parsons, J. Paul, M. J. Rees, C. S. Reynolds, J. E. Rhoads, T. P. Sasseen, B. E. Schaefer, A. T. Short, A. P. Smale, I. A. Smith, L. Stella, G. Tagliaferri, T. Takahashi, M. Tashiro, L. K. Townsley, J. Tueller, M. J. L. Turner, M. Vietri, W. Voges, M. J. Ward, R. Willingale, F. M. Zerbi, and W. W. Zhang. The Swift Gamma-Ray Burst Mission. ApJ, 611:1005–1020, August 2004. doi:10.1086/422091.

P. Graff, F. Feroz, M. P. Hobson, and A. Lasenby. BAMBI: blind accelerated multimodal Bayesian inference. MNRAS, 421:169–180, March 2012. doi:10.1111/j.1365-2966.2011.20288.x.

P. Graff, F. Feroz, M. P. Hobson, and A. Lasenby. SKYNET: an efficient and robust neural network training tool for machine learning in astronomy. MNRAS, 441:1741–1759, June 2014. doi:10.1093/mnras/stu642.

D. Guetta and M. Della Valle. On the Rates of Gamma-Ray Bursts and Type Ib/c Supernovae. ApJ, 657:L73–L76, March 2007. doi:10.1086/511417.

D. Guetta and T. Piran. Do long duration gamma ray bursts follow star formation? J. Cosmology Astropart. Phys., 7:003, July 2007. doi:10.1088/1475-7516/2007/07/003.

Andrew M. Hopkins and John F. Beacom. On the normalization of the cosmic star formation history. *The Astrophysical Journal*, 651(1):142, 2006. URL http://stacks.iop.org/0004-637X/651/i=1/a=142.

E. J. Howell, D. M. Coward, G. Stratta, B. Gendre, and H. Zhou. Constraining the rate and luminosity function of Swift gamma-ray bursts. MNRAS, 444:15–28, October 2014. doi:10.1093/mnras/stu1403.

E. E. O. Ishida, R. S. de Souza, and A. Ferrara. Probing cosmic star formation up to z= 9.4 with gamma-ray bursts. MNRAS, 418:500–504, November 2011. doi:10.1111/j.1365-2966.2011.19501.x.

P. Jakobsson, J. Hjorth, D. Malesani, R. Chapman, J. P. U. Fynbo, N. R. Tanvir, B. Milvang-Jensen, P. M. Vreeswijk, G. Letawe, and R. L. C. Starling. The Optically Unbiased GRB Host (TOUGH) Survey. III. Redshift Distribution. ApJ, 752:62, June 2012. doi:10.1088/0004-637X/752/1/62.

C. Kanaan and J. A. de Freitas Pacheco. Revisiting the formation rate and redshift distribution of long gamma-ray bursts. A&A, 559:A64, November 2013. doi:10.1051/0004-6361/201321963.

M. D. Kistler, H. Yüksel, J. F. Beacom, and K. Z. Stanek. An Unexpectedly Swift Rise in the Gamma-Ray Burst Rate. ApJ, 673:L119–L122, February 2008. doi:10.1086/527671.

M. D. Kistler, H. Yüksel, J. F. Beacom, A. M. Hopkins, and J. S. B. Wyithe. The Star Formation Rate in the Reionization Era as Indicated by Gamma-Ray Bursts. ApJ, 705:L104–L108, November 2009. doi:10.1088/0004-637X/705/2/L104.

T. Le and C. D. Dermer. On the Redshift Distribution of Gamma-Ray Bursts in the Swift Era. ApJ, 661:394–415, May 2007. doi:10.1086/513460.

A. Lien, T. Sakamoto, N. Gehrels, D. M. Palmer, S. D. Barthelmy, C. Graziani, and J. K. Cannizzo. Probing the Cosmic Gamma-Ray Burst Rate with Trigger Simulations of the Swift Burst Alert Telescope. ApJ, 783:24, March 2014. doi:10.1088/0004-637X/783/1/24.

David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. www.inference.phy.cam.ac.uk/mackay/itila/.

K. M. McLean, E. E. Fenimore, D. Palmer, S. Barthelmy, N. Gehrels, H. Krimm, C. Markwardt, and A. Parsons. Setting the Triggering Thresholds on Swift. In E. Fenimore and M. Galassi, editors, *Gamma-Ray Bursts: 30 Years of Discovery*, volume 727 of *American Institute of Physics Conference Series*, pages 667–670, September 2004. doi:10.1063/1.1810931.

D. M. Palmer, E. Fenimore, M. Galassi, K. McLean, T. Tavenner, S. Barthelmy, M. Blau, J. Cummings, N. Gehrels, D. Hullinger, H. Krimm, C. Markwardt, R. Mason, J. Ong, J. Polk, A. Parsons, L. Shackelford, J. Tueller, S. Walling, Y. Okada, H. Takahashi, M. Toshiro, M. Suzuki, G. Sato, T. Takahashi, and S. Watanabe. The BAT-Swift Science Software. In E. Fenimore and M. Galassi, editors, *Gamma-Ray Bursts: 30 Years of Discovery*, volume 727 of *American Institute of Physics Conference Series*, pages 663–666, September 2004. doi:10.1063/1.1810930.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL http://www.jmlr.org/papers/v12/pedregosa11a.html.

A. Pélangeon, J.-L. Atteia, Y. E. Nakagawa, K. Hurley, A. Yoshida, R. Vanderspek, M. Suzuki, N. Kawai, G. Pizzichini, M. Boër, J. Braga, G. Crew, T. Q. Donaghy, J. P. Dezalay, J. Doty, E. E. Fenimore, M. Galassi, C. Graziani, J. G. Jernigan, D. Q. Lamb, A. Levine, J. Manchanda, F. Martel, M. Matsuoka, J.-F. Olive, G. Prigozhin, G. R. Ricker, T. Sakamoto, Y. Shirasaki, S. Sugita, K. Takagishi, T. Tamagawa, J. Villasenor, S. E. Woosley, and M. Yamauchi. Intrinsic properties of a complete sample of HETE-2 gamma-ray bursts. A measure of the GRB rate in the Local Universe. A&A, 491:157–171, November 2008. doi:10.1051/0004-6361:200809709.

A. Pescalli, G. Ghirlanda, R. Salvaterra, G. Ghisellini, S. D. Vergani, F. Nappo, O. S. Salafia, A. Melandri, S. Covino, and D. Götz. The rate and luminosity function of long Gamma Ray Bursts. *ArXiv e-prints*, June 2015.

V. Petrosian, E. Kitanidis, and D. Kocevski. Cosmological Evolution of Long Gamma-Ray Bursts and the Star Formation Rate. ApJ, 806:44, June 2015. doi:10.1088/0004-637X/806/1/44.

S.-F. Qin, E.-W. Liang, R.-J. Lu, J.-Y. Wei, and S.-N. Zhang. Simulations on high-z long gamma-ray burst rate. MNRAS, 406:558–565, July 2010. doi:10.1111/j.1365-2966.2010.16691.x.

B. E. Robertson and R. S. Ellis. Connecting the Gamma Ray Burst Rate and the Cosmic Star Formation History: Implications for Reionization and Galaxy Evolution. ApJ, 744:95, January 2012. doi:10.1088/0004-637X/744/2/95.

R. Salvaterra, C. Guidorzi, S. Campana, G. Chincarini, and G. Tagliaferri. Evidence for luminosity evolution of long gamma-ray bursts in Swift data. MNRAS, 396:299–303, June 2009. doi:10.1111/j.1365-2966.2008.14343.x.

R. Salvaterra, S. Campana, S. D. Vergani, S. Covino, P. D'Avanzo, D. Fugazza, G. Ghirlanda, G. Ghisellini, A. Melandri, L. Nava, B. Sbarufatti, H. Flores, S. Piranomonte, and G. Tagliaferri. A Complete Sample of Bright Swift Long Gamma-Ray Bursts. I. Sample Presentation, Luminosity Function and Evolution. ApJ, 749:68, April 2012. doi:10.1088/0004-637X/749/1/68.

S. Stevenson, F. Ohme, and S. Fairhurst. Distinguishing compact binary population synthesis models using gravitational-wave observations of coalescing binary black holes. *ArXiv e-prints*, April 2015.

N. R. Tanvir, A. J. Levan, A. S. Fruchter, J. P. U. Fynbo, J. Hjorth, K. Wiersema, M. N. Bremer, J. Rhoads, P. Jakobsson, P. T. O'Brien, E. R. Stanway, D. Bersier, P. Natarajan, J. Greiner, D. Watson, A. J. Castro-Tirado, R. A. M. J. Wijers, R. L. C. Starling, K. Misra, J. F. Graham, and C. Kouveliotou. Star Formation in the Early Universe: Beyond the Tip of the Iceberg. ApJ, 754:46, July 2012. doi:10.1088/0004-637X/754/1/46.

F. J. Virgili, B. Zhang, K. Nagamine, and J.-H. Choi. Gamma-ray burst rate: high-redshift excess and its possible origins. MNRAS, 417:3025–3034, November 2011. doi:10.1111/j.1365-2966.2011.19459.x.

D. Wanderman and T. Piran. The luminosity function and the rate of Swift's gamma-ray bursts. MNRAS, 406:1944–1958, August 2010. doi:10.1111/j.1365-2966.2010.16787.x.

F. Y. Wang. The high-redshift star formation rate derived from gamma-ray bursts: possible origin and cosmic reionization. A&A, 556:A90, August 2013. doi:10.1051/0004-6361/201321623.

S. E. Woosley and A. Heger. Long Gamma-Ray Transients from Collapsars. ApJ, 752:32, June 2012. doi:10.1088/0004-637X/752/1/32.

H. Yu, F. Y. Wang, Z. G. Dai, and K. S. Cheng. An Unexpectedly Low-redshift Excess of Swift Gamma-ray Burst Rate. ApJS, 218:13, May 2015. doi:10.1088/0067-0049/218/1/13.

H. Yüksel, M. D. Kistler, J. F. Beacom, and A. M. Hopkins. Revealing the High-Redshift Star Formation Rate with Gamma-Ray Bursts. ApJ, 683: L5–L8, August 2008. doi:10.1086/591449.