Rehemtulla, N, Miller, AA, Jegou Du Laz, T, Coughlin, MW, Fremling, C, Perley, DA, Qin, YJ, Sollerman, J, Mahabal, AA, Laher, RR, Riddle, R, Rusholme, B and Kulkarni, SR

The Zwicky Transient Facility Bright Transient Survey. III. BTSbot: Automated Identification and Follow-up of Bright Transients with Deep Learning

The Astrophysical Journal, 972:7 (19pp), 2024 September 1 https://doi.org/10.3847/1538-4357/ad5666

The Zwicky Transient Facility Bright Transient Survey. III. BTSbot: Automated Identification and Follow-up of Bright Transients with Deep Learning

Nabeel Rehemtulla[1,2], Adam A. Miller[1,2], Theophile Jegou Du Laz[3], Michael W. Coughlin[4], Christoffer Fremling[3,5], Daniel A. Perley[6], Yu-Jing Qin[3], Jesper Sollerman[7], Ashish A. Mahabal[3,8], Russ R. Laher[9], Reed Riddle[5], Ben Rusholme[9], and Shrinivas R. Kulkarni[3]

[1] Department of Physics and Astronomy, Northwestern University, 2145 Sheridan Rd., Evanston, IL 60208, USA; nabeelr@u.northwestern.edu
[2] Center for Interdisciplinary Exploration and Research in Astrophysics (CIERA), 1800 Sherman Ave., Evanston, IL 60201, USA
[3] Division of Physics, Mathematics, and Astronomy 249-17, California Institute of Technology, Pasadena, CA 91125, USA
[4] School of Physics and Astronomy, University of Minnesota, Minneapolis, MN 55455, USA
[5] Caltech Optical Observatories, California Institute of Technology, Pasadena, CA 91125, USA
[6] Astrophysics Research Institute, Liverpool John Moores University, IC2, Liverpool Science Park, 146 Brownlow Hill, Liverpool L3 5RF, UK
[7] Department of Astronomy, The Oskar Klein Center, Stockholm University, AlbaNova, SE-10691 Stockholm, Sweden
[8] Center for Data Driven Discovery, California Institute of Technology, Pasadena, CA 91125, USA
[9] IPAC, California Institute of Technology, 1200 E. California Blvd., Pasadena, CA 91125, USA

Received 2024 January 29; revised 2024 May 28; accepted 2024 May 31; published 2024 August 21

Abstract

The Bright Transient Survey (BTS) aims to obtain a classification spectrum for all bright ($m_{\rm peak} \leqslant 18.5$ mag) extragalactic transients found in the Zwicky Transient Facility (ZTF) public survey. BTS critically relies on visual inspection ("scanning") to select targets for spectroscopic follow-up, which, while effective, has required a significant time investment over the past ~5 yr of ZTF operations. We present `BTSbot`, a multimodal convolutional neural network, which provides a bright transient score to individual ZTF detections using their image data and 25 extracted features. `BTSbot` is able to eliminate the need for daily human scanning by automatically identifying and requesting spectroscopic follow-up observations of new bright transient candidates. `BTSbot` recovers all bright transients in our test split and performs on par with scanners in terms of identification speed (on average, ~1 hr quicker than scanners). We also find that `BTSbot` is not significantly impacted by any data shift by comparing performance across a concealed test split and a sample of very recent BTS candidates. `BTSbot` has been integrated into `Fritz` and `Kowalski`, ZTF's first-party marshal and alert broker, and now sends automatic spectroscopic follow-up requests for the new transients it identifies. Between 2023 December and 2024 May, `BTSbot` selected 609 sources in real time, 96% of which were real extragalactic transients. With `BTSbot` and other automation tools, the BTS workflow has produced the first fully automatic end-to-end discovery and classification of a transient, representing a significant reduction in the human time needed to scan.

Unified Astronomy Thesaurus concepts: Time domain astronomy (2109); Sky surveys (1464); Supernovae (1668); Convolutional neural networks (1938)

1. Introduction

Large, wide-field surveys like the Panoramic Survey Telescope and Rapid Response System (Pan-STARRS; Kaiser et al. 2002), the All-Sky Automated Survey for Supernovae (Shappee et al. 2014), the Asteroid Terrestrial Last-Alert System (Tonry 2011; Tonry et al. 2018; Smith et al. 2020), and the Zwicky Transient Facility (ZTF; Bellm et al. 2019a, 2019b; Graham et al. 2019; Masci et al. 2019; Dekany et al. 2020) have made immense contributions to time-domain astronomy by repeatedly imaging the entire night sky. Some surveys interface with the community through the output of an alert stream. Alert packets comprising the stream are intended to notify the community of the statistically significant brightening or dimming of some source with respect to a historical reference image. The alert stream provides a time series for a wide range of astrophysical, and nonastrophysical, phenomena. We are interested in the study of supernovae (SNe) using these alert streams, although only a small fraction of alerts in the unfiltered stream originate from genuine SNe. Thus, alert filters are deployed to identify candidate sources of interest. It is practically impossible to design an alert filter that rejects every irrelevant alert and accepts all alerts related to sources of interest. Therefore, manual candidate vetting, or "scanning," of the filtered alerts is required. The term scanning can refer to different actions depending on the relevant survey and science case, but, in general, scanning has been used extensively and very successfully in the SN community to identify SNe for follow-up observations and further study.

The nightly data rate of the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST; Ivezić et al. 2019) will significantly surpass our collective capacity for human scanning. Previous and ongoing surveys employ machine learning (ML) techniques for real/bogus classification (e.g., Bailey et al. 2007; Bloom et al. 2012; Brink et al. 2013; Goldstein et al. 2015; Wright et al. 2015; Cabrera-Vives et al. 2017; Duev et al. 2019; Mahabal et al. 2019; Turpin et al. 2020; Killestein et al. 2021), and adopting ML will be near compulsory to efficiently extract knowledge from the next generation of surveys.

Beyond real/bogus classification, ML models have also been applied to a variety of tasks in astronomy including photometric transient classification (e.g., Boone 2019;

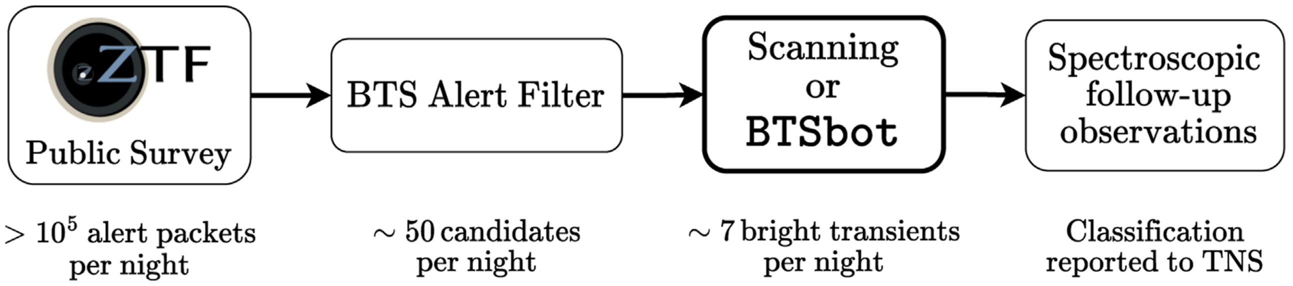

ZTF
Public Survey
BTS Alert Filter
Scanning or BTSbot
Spectroscopic follow-up observations
$> 10^5$ alert packets per night
~ 50 candidates per night
~ 7 bright transients per night
Classification reported to TNS

Figure 1. Diagram of the BTS workflow. The BTS alert filter ingests the ZTF public survey alert stream and, using upstream tools like `braai` and `sgscore`, removes bogus alerts, dim sources, and sources that are trivially not bright transients. Selection of real bright transients from the pool of candidates is done by visual inspection ("scanning"); in this study, we develop an ML-based alternative: `BTSbot`. The selected bright transients receive spectroscopic follow-up and classification, and are promptly reported to the public.

Muthukrishna et al. 2019; Villar et al. 2019, 2020; Hosseinzadeh et al. 2020; Möller & de Boissière 2020; Qu et al. 2021; Gagliano et al. 2023; de Soto et al. 2024), photometric redshift estimation (e.g., Carrasco Kind & Brunner 2013; Sadeh et al. 2016; Pasquet et al. 2019), and many others. Some have also been developed to, in part, reduce the scanning load by encoding phenomenological or taxonomic information of a source in their model's output (e.g., Bailey et al. 2007; Gomez et al. 2020, 2023; Carrasco-Davis et al. 2021; Duev & van der Walt 2021; Stein et al. 2024).

For the most part, ML models in astronomy perform their tasks using extracted numeric features with architectures like random forests (Breiman 2001), or fully connected neural networks (NNs; McCulloch & Pitts 1943; LeCun et al. 2015). While appropriate in some cases, limiting these models to extracted features alone ignores potentially valuable information present in the images from which the features are extracted. A comparatively small number of convolutional neural networks (CNNs; Fukushima & Miyake 1982) have been built that make use of the information embedded in astronomical images, and they have generally had great success (e.g., Dieleman et al. 2015; Domínguez Sánchez et al. 2018; Lanusse et al. 2018; Duev et al. 2019; Walmsley et al. 2020). CNNs are particularly well suited to astronomy because they can capture properties, like galaxy morphology, which often remain largely obscured to other image-processing techniques (Walmsley et al. 2019). Only a very small subset of these CNNs are *multimodal*, meaning they take in images and input of another type, like extracted features or a light curve (e.g., Carrasco-Davis et al. 2021; Duev & van der Walt 2021; van Roestel et al. 2021; Morgan et al. 2022, 2023; Stoppa et al. 2023). We have developed a multimodal CNN (MM-CNN) to automate scanning for a filtered stream from ZTF.

The Bright Transient Survey (BTS; Fremling et al. 2020; Perley et al. 2020) aims to spectroscopically classify all bright ($m_{\rm peak} \leqslant 18.5$ mag in g or r band) extragalactic transients[10] from the ZTF public alert stream (Patterson et al. 2019). The ZTF public survey produces $>10^5$ alert packets per night (Mahabal et al. 2019). The majority of these are from variable sources or are bogus alerts, those arising from nonastrophysical phenomena. The BTS alert filter (Perley et al. 2020) removes from consideration most bogus alerts and many alerts from asteroids, active galactic nuclei (AGN), cataclysmic variables (CVs), variable stars (VarStars), and alerts from sources with $m > 19.0$ mag in g and r band. This filter makes use of `braai` (Duev et al. 2019) and `sgscore` (Tachibana & Miller 2018) to filter the stream down to ~50 new candidate BTS sources per night, of which ~7 are new real bright transients. The other BTS candidates are mostly dim ($m_{\rm peak} > 18.5$ mag) SNe or AGN, CVs, and VarStars that were not removed by the alert filter. All nightly BTS candidates are scanned by experts ("scanners") who catalog ("save") the real bright transients and request spectroscopic observations for classification. Scanning is performed on `Fritz`,[11] ZTF's first-party marshal and a SkyPortal instance (van der Walt et al. 2019; Coughlin et al. 2023). BTS primarily executes its follow-up observations with robotic spectrographs like the spectral energy distribution (SED) machine (SEDM; Blagorodnova et al. 2018; Rigault et al. 2019; Kim et al. 2022), and, soon, the SED machine Kitt Peak[12] (SEDM-KP). The resulting classifications, whether assigned automatically by `SNIascore` (Fremling et al. 2021) or manually by a scanner, are promptly reported to the public via the Transient Name Server[13] (TNS), as visualized in Figure 1.

The BTS catalog is available online[14] and is updated in real-time. Since its origin in 2018 May, BTS has maintained exceptionally high spectroscopic completeness of relevant sources (95.4%[15]), and BTS serves the community by rapidly releasing their classifications to the public. As of 2023 October, the BTS sample contains more than 8300 publicly classified SNe.

BTS enables a large amount of science, notably including some of the largest SN population studies conducted to date (e.g., Perley et al. 2020; Irani et al. 2022; Sharon & Kushnir 2022; Sollerman et al. 2022; Cold & Hjorth 2023; Rodríguez et al. 2023; Sharma et al. 2023). The survey also provides unique discoveries (e.g., Yang et al. 2021; Goobar et al. 2023) and is paving the way for using SNe to study large scale structure (Tsaprazi et al. 2022).

Our new model, `BTSbot`, enables the automation of scanning and spectroscopic follow-up for BTS by performing binary classification: bright transient/not bright transient. `BTSbot` produces a unit-interval bright transient score for an input ZTF Avro[16] alert packet (Masci et al. 2019; Patterson et al. 2019) augmented with some custom metadata features. It

[10] Henceforth, we use "supernova" and "transient" interchangeably with "extragalactic transient." These are not equivalent, but it simplifies the prose.

[11] https://github.com/fritz-marshal/fritz

[12] https://sites.astro.caltech.edu/sedmkp/

[13] https://www.wis-tns.org

[14] https://sites.astro.caltech.edu/ztf/bts

[15] Computed for sources passing the BTS alert filter with $m_{\rm peak} \leqslant 18.5$ mag, good light-curve coverage, and displaying an SN-like light curve or a galaxy crossmatch from 2018 May to 2023 September. We adopt this value for the BTS sample completeness henceforth. See Perley et al. (2020) for detailed definition of sample cuts.

[16] https://avro.apache.org

is trained to run only on alerts from sources with at least one alert passing the BTS alert filter (candidate BTS sources), the same alerts that are provided to scanners. From image cutouts and extracted features, `BTSbot` must simultaneously separate transients from variable sources and, while $m > 18.5$ mag, predict whether or not the source will attain $m_{\mathrm{peak}} \leqslant 18.5$ mag.

`BTSbot` is now integrated into `Kowalski` (Duev et al. 2019; Coughlin et al. 2023), ZTF's first-party alert broker, and `Fritz`, where `BTSbot` has now entered the BTS workflow. `BTSbot` is able to automatically save sources to BTS catalogs and conditionally trigger SEDM/SEDM-KP. `BTSbot` joins a rich collection of ML models and automation tools central to daily BTS operations, namely `braai` (Duev et al. 2019), `sgscore` (Tachibana & Miller 2018), `pySEDM` (Rigault et al. 2019), and `SNIascore` (Fremling et al. 2021). Together, this workflow has yielded the first transient to be fully automatically detected, identified, spectroscopically classified, and publicly reported: SN 2023tyk (Rehemtulla et al. 2023a). Zero human action was involved from the first detection to the publicly reported spectroscopic classification. Combining complementary ML models allows for the automation of a significant fraction of the tasks necessary to maintain BTS. This has the side effect of freeing up time otherwise spent on repetitive, well-understood tasks, allowing for the better allocation of expert time and resources. We make the latest `BTSbot` source code and trained model publicly available on GitHub[17] and the version described in this publication available on Zenodo (Rehemtulla et al. 2024).

2. Training Data

The quality of an ML model's training set is a key factor in determining its performance. ML models, especially deep learning models like `BTSbot`, are known to behave unpredictably when exposed to data unlike what they were trained on (Szegedy et al. 2013; Hendrycks & Gimpel 2016). Thus, a model's training set must be fully representative of the model's input domain. `BTSbot`'s domain is ZTF alert packets from BTS candidates. These alerts come from SNe, AGN, CVs, VarStars, novae in very nearby systems, and a small number of other miscellaneous events including bogus alerts of many types (e.g., those due to poor image subtractions, high-proper-motion stars, saturated stars, etc.). We compile an extensive list of ZTF identifiers (ZTF-IDs) for sources that fall into these categories by drawing from a number of repositories. With these selections, we fully represent the wide variety of astrophysical phenomena `BTSbot` is exposed to in production.

An internal version of the BTS Sample Explorer is the most significant contributor to our initial list of ZTF-IDs. This internal BTS Sample Explorer operates identically to the public-facing version presented in Appendix F of Perley et al. (2020), but it adds information from internal ZTF catalogs. One of these catalogs is assembled by scanning an alert filter nearly identical to the BTS alert filter but altered to include sources with $19 < m_{\mathrm{peak}}\,[\mathrm{mag}] < 19.8$. Although these are not strictly BTS candidates, we allow them into the training set to increase the number of faint alerts.

We draw from these catalogs with three queries. The first query, "*trues*" for short, selects bright ($m_{\mathrm{peak}} \leqslant 18.5$ mag) spectroscopically confirmed extragalactic transients that pass the purity cut (i.e., have a galaxy crossmatch or an SN-like light curve; see Section 2.4 of Perley et al. 2020 for details). These bright confirmed transients make up the entire positive/true class of our training set. The second query, "*vars*" for short, selects any source classified as an AGN, CV, or quasar (QSO) in any of the internal BTS catalogs. These sources are all considered nontransients or are not extragalactic and thus are part of the negative/false class regardless of their peak magnitude. The third query, "*dims*" for short, selects any source that passes the purity cut with pre- and post-peak light-curve coverage (see Section 2.3 of Perley et al. 2020 for coverage definitions) and $m_{\mathrm{peak}} > 18.5$ mag. This is designed to broadly select dim SNe, which, because of their peak magnitudes, are part of the negative class. The *dims* query is dominated by SNe but also includes a small number of other sources, e.g., tidal disruption events, CVs, asteroids, VarStars, etc. The requirement of pre- and post-peak light-curve coverage is added to avoid selecting bright SNe with poor photometric coverage near peak, which could then appear as having dimmer peak magnitudes. Because peak magnitudes presented on the BTS sample explorer are computed from public data alone, there is a small population of sources that, due to alerts from partnership data, have $m_{\mathrm{peak}} \leqslant 18.5$ mag but appear in *dims*. These are all removed entirely from the training set because many of them are unclassified.[18] By definition, neither *vars* nor *dims* overlap with *trues*, but the imperfect nature of the selections done for *vars* and *dims* creates some overlap between the two. Sources in the overlap are removed from *dims* and kept only in *vars*.

Any source present on the BTS Sample Explorer was once marked as a bright transient by a human scanner. While we do build a sizeable list of non-BTS sources saved by the human scanners, there is a missing population of BTS candidates that are never saved, "*rejects*" for short. Our compilation of *rejects* is limited to sources with alerts that pass the BTS alert filter between 2021 January 1 and 2023 January 1 UTC.[19] Many candidates from before 2021 January 1 encountered earlier versions of the BTS alert filter, which was last improved in late 2020; many candidates from after 2023 January 1 were still evolving at the time the sources were queried and thus have potentially uncertain types. By not saving them, scanners implicitly mark these sources as non-BTS sources and thus part of our negative class. This assumption is reasonable given BTS's very high photometric completeness. An extremely small fraction of the sources in the *rejects* list may be bright transients and inject a small amount of label noise into our training set, but we find that this effect, if present at all, is small enough to be ignored.

There is an additional population of BTS candidates, which we do not include in our training set: "*junk*" for short. These are sources that frequently pass the BTS alert filter but are marked as *junk* by scanners because they are clearly not bright transients. They tend to be very long-lived sources, like AGN or VarStars, but also include many high-proper-motion stars and bogus sources. They are removed from the training set for two reasons: (i) experiments that included *junk* sources consistently yielded worse performance than equivalent models that excluded them; and (ii) in practice, sources cataloged as *junk* are typically automatically rejected and not presented to scanners. Although scanners only inconsistently tag sources as

[17] https://github.com/nabeelre/BTSbot

[18] These were identified after the test split was revealed, so, to preserve the same splits, they are removed after train/validation/test splitting is performed.

[19] All dates and times here are in UTC unless otherwise specified.

Table 1
Training Set Size before/after Cleaning Cuts

Name of Query	Number of Sources	Number of Alerts
Initial Queries		
trues[a]	5212	308,934
vars[b]	1127	150,017
dims[c]	8979	249,087
rejects[d]	4417	407,357
Total	19,735	1,115,395
Cleaned Training Set		
trues[a]	5206	264,317
vars[b]	1126	109,934
dims[c]	8824	223,934
rejects[d]	4402	241,478
Total	19,558	839,663

Notes. Alerts are removed from the training set if they (i) have a corrupted image cutout, (ii) come from a source with an ambiguous label, (iii) are missing Pan-STARRS1 crossmatch information, (iv) are an *i*-band observation, (v) have a negative difference image, or (vi) come from a source with a transient present in the reference image.
[a] Spectroscopically confirmed bright ($m_{\rm peak} \leqslant 18.5$ mag) extragalactic transients.
[b] Sources classified as AGN, CVs, VarStars, or QSOs.
[c] Dim ($m_{\rm peak} > 18.5$ mag) sources with transient-like light curves.
[d] Sources not marked as bright extragalactic transients by BTS scanners.

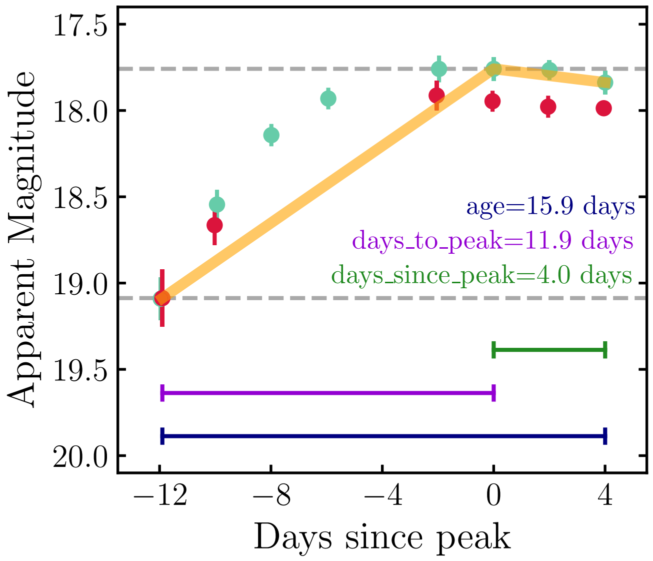


Figure 2. Custom metadata feature definitions depicted for the light curve of ZTF20acjlkpe. Teal (*g* band) and red (*r* band) circles indicate detections. `days_to_peak` (purple), `days_since_peak` (green), `age` (navy), `peakmag_so_far` (upper dashed gray), and `maxmag_so_far` (lower dashed gray) are presented for the latest detection shown. Together, these features make simplified information of the light-curve phase and shape available to `BTSbot` (shown as the orange line, which assumes that `maxmag_so_far` corresponds to the first detection).

junk, removing these sources is effectively a stage of prefiltering similar to how the BTS alert filter removes alerts trivially irrelevant to BTS. Some sources selected by our other queries, like AGN in *vars*, are also cataloged in *junk*; these sources remain in the other query and are not removed with other *junk* sources. This configuration allows for some confidently identified variable sources to be included in the training set, while rejecting other nuisance sources.

Once we have a list of ZTF-IDs and their corresponding labels, we query `Kowalski` to retrieve all alert packets from each source. BTS only uses data from the ZTF public survey, but we include ZTF partnership data to increase the size of our training set. We arrange the science, reference, and difference image cutouts of each alert packet to form a $63 \times 63 \times 3$ image, or a triplet. Image cutouts are individually normalized with Euclidean normalization[20]; pixels in masked regions are uniformly given values of 0; cutouts smaller than 63×63 (due to being near the edge of a CCD) are padded to 63×63 with pixels of a value of 10^{-9}. This image preprocessing is identical to `braai`'s (Duev et al. 2019). Very rarely (∼0.4% of alerts), queried cutouts will have pixels with values of -3.4×10^{38} (the minimum value for a `float32` object). In order to remain consistent with the image processing in `braai`, we do not mask these pixels. Instead, we consider these cutouts corrupted, and alerts with any corrupted cutouts are removed. The distribution of alerts and sources in the training set at this stage is shown in Table 1 under the initial queries header.

We then augment the alert packets with custom metadata features that are not already present in the ZTF Avro alert packets: `days_to_peak`, `days_since_peak`, `age`, `peakmag_so_far`, `maxmag_so_far`, and `nnondet`. The feature `days_to_peak` encodes the number of days from the first alert to the alert with the brightest magnitude thus far for the source in question. Closely related is `days_since_peak`, which represents the number of days from the current alert to the alert with the brightest magnitude. The sum of these quantities is the `age`. The features `peakmag_so_far` and `maxmag_so_far` encode the brightest and dimmest magnitude of all alert packets from this source thus far; `peakmag_so_far` is particularly crucial for correctly classifying late-time alerts of SNe because `BTSbot` would otherwise have almost no information on the brightness history of the source. These features are shown over an example light curve in Figure 2. They clearly do not recover the light curve perfectly, but they are adopted because of their simplicity and effectiveness across a very large variety of light curves. We find that giving `BTSbot` high-level information on the source's current phase and the rough shape of its light curve with these features yields a significant boost in performance. Many methods exist to more faithfully represent SNe light curves, but many are not conditioned to model light curves of the other sources in `BTSbot`'s domain, e.g., AGN, VarStars, and others. Notably, these features are defined for all sources with as few as one detection or detections in only one photometric passband. One could extract more information from the light curve by decoupling these features to passband-dependent equivalents. This would, however, create instances of alerts having missing features, requiring the adoption of some placeholder value like −999. We discuss our reasons for not including features that would require such placeholder values later in this section. Lastly, the feature `nnondet` is an estimation of the number of nondetections at the source's location. This feature was used in the Automatic Learning for the Rapid Classification of Events (ALeRCE) real-time stamp classifier (Carrasco-Davis et al. 2021) and was found to have high importance for distinguishing alerts between their five classes; we find that including it similarly boosts `BTSbot`'s performance.

[20] Also called L^2 normalization.

All of these features are computed for each alert packet from its perspective, i.e., only using information already available at the time the alert packet was created, including alerts from before each source passed the BTS alert filter. Additionally, some alerts were created before the latest version of `braai` went into production and thus have a deep real-bogus score (`drb`) from an outdated version of `braai` or no `drb` at all. We rerun all alert packets through the `d6_m9` version of `braai` and replace all `drb` scores with these new scores.

Next, we clean our training set with a series of cuts. First, we remove alerts from *i*-band observations and alerts with negative difference images. The *i*-band alerts are removed primarily because BTS draws from the ZTF public survey, which only takes observations in *g* and *r* band. Negative difference images are identified using the `isdiffpos` flag included in the ZTF alert packets. Alerts with negative difference images are removed because they frequently represent instances of SNe or other transients visible in the reference image: a configuration different than what is typical for an alert. As an extra precaution, we compile a list of sources with SNe present in the reference images and remove them from the training set. Additionally, some sources fit into neither or both of our classes and are correspondingly removed from our training set. For example, ZTF18abdiasx appears to be a bright SN projected on top of an AGN, and thus fits in both of our classes, so it is removed from the training set. Some of the metadata features we choose to include reference the Pan-STARRS1 catalog (PS1; Kaiser et al. 2002): `sgscore{1,2}` and `distpsnr{1,2}`. When there are no PS1 sources within 30″ of a given ZTF alert, these features are set to −999 by the upstream ZTF pipeline. We run experiments including/excluding examples with −999 in any of these fields and find that `BTSbot` tends to perform better when not fed examples with such values. It is for this reason that we remove all alerts with −999 in any of these fields and choose to omit other features that would require similar placeholder values, e.g., passband-dependent light-curve features.

Table 1 shows the counts of sources and alerts at this stage under the cleaned training set header. Each of the four queries loses some alerts in the cleaning process; however, the *rejects* query loses more than 150,000 alerts. These 150,000 alerts are almost entirely negative difference image alerts, which come from sources like binary stars, AGN, and high-proper-motion stars.

We employ standard practices for splitting the full training set into train, validation, and test splits. We randomly assign sources to these splits with probabilities of 81%, 9%, and 10% respectively.[21] This splitting is done on sources rather than alerts to prevent any source from having alerts in multiple splits, which would produce a validation and test bias that overestimates the true performance of the model.

Figure 3 illustrates that, as expected, the typical number of alerts per source is variable over the four different queries. Some AGN in *vars* have $>10^3$ alerts, while some SNe have as few as two alerts. We correct this imbalance by defining a hyperparameter $N_{\rm max}$: the maximum number of alerts per source allowed in the training set. Sources with more than $N_{\rm max}$ alerts will have some alerts removed, and those with $N_{\rm max}$ or fewer alerts will remain untouched.

[21] 90% of the training data is seen by the model during development, and the remaining 10% is concealed until hyperparameters are finalized.

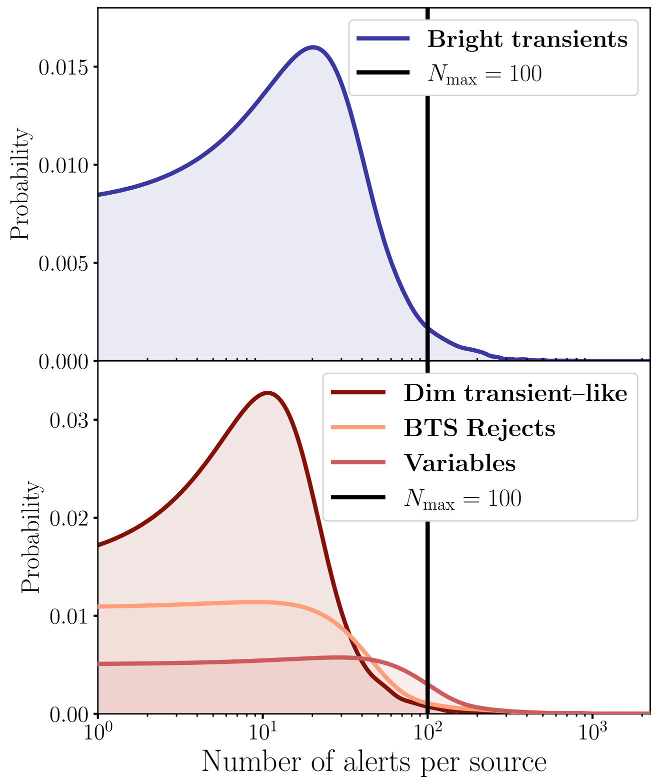


Figure 3. Gaussian kernel density estimations showing the number of alerts per source for each query comprising our cleaned training set. Top: positive class examples. Bottom: negative class examples. Sources with many alerts are thinned down to $N_{\rm max} = 100$ alerts per source. This prevents long-lived sources like AGN (in *vars*) and bright SNe (in *trues*) from being overrepresented in training.

We apply $N_{\rm max}$ alert thinning to sources based on which query they originate from and which split they are present in. For sources in the *trues*, *dims*, and *rejects* queries (many of which are galactic or extragalactic transients), we want `BTSbot` to learn and identify their properties at all phases of their evolution, so we take a simple uniform random selection of $N_{\rm max}$ alerts to preserve and discard all others. The remaining alerts are roughly uniformly distributed over the rise, peak, and fade of these transients.[22] For sources in the *vars* query, we keep only the most recent $N_{\rm max}$ alerts. Metadata of long-lived sources look systematically different in the present than they did years ago. For example, AGN were typically first detected near the beginning of ZTF and have been repeatedly detected since. New incoming alerts from these AGN, those that `BTSbot` will encounter in production, will typically have very large `age` and `ndethist` (approximately, number of previous detections), but those nearer to when they were first detected will have much smaller `age` and `ndethist`. By selecting the latest $N_{\rm max}$ alerts, we capture those that are most representative of today's BTS candidates, and we avoid a data shift that would dramatically worsen `BTSbot`'s performance in production. Early versions of `BTSbot` used random $N_{\rm max}$ thinning on *vars* sources and yielded markedly worse performance. We find the optimal value to be $N_{\rm max} = 100$ (see Section 3.1 for optimization details).

[22] Observational coverage of transients is typically greater near peak, so this simple random selection will slightly favor removing alerts near peak. We do not expect significantly different results if we attempted to correct for this.

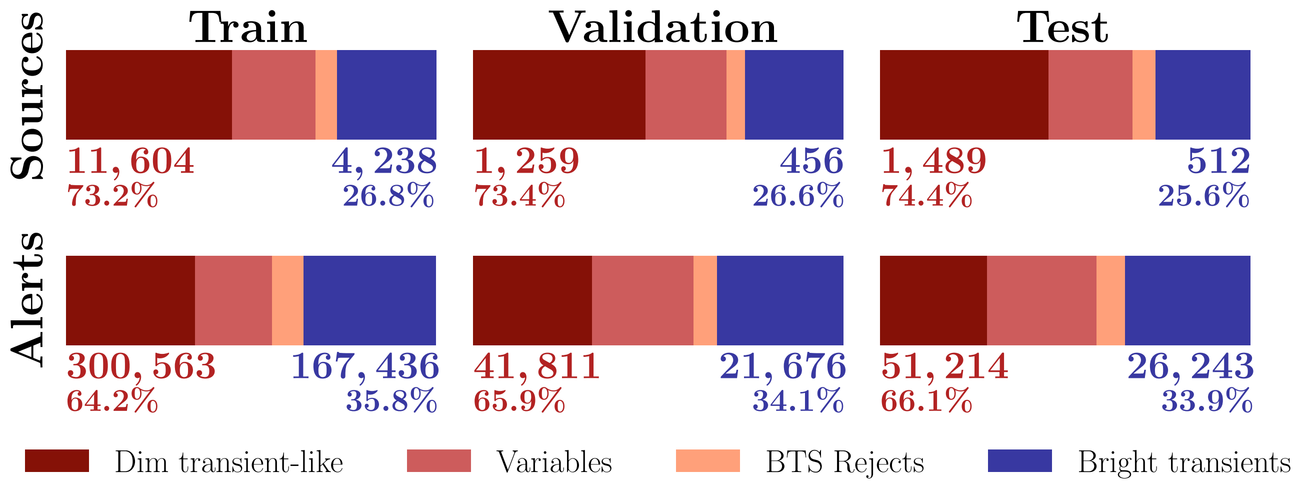


Figure 4. Bar charts showing the distribution of sources and alerts from four queries into train/validation/test (81%/9%/10%) splits. Certain sources have an upper limit set on their number of alerts; excess alerts are removed. Our training set is imbalanced favoring not-BTS sources ∼3:1 and favoring the not-BTS alerts ∼2:1.

The CVs in *vars* are transients like the sources in *trues* and *dims*, and we want to capture alerts from all stages of their evolution, rather than just late-time. The selection of the latest $N_{\max} = 100$ alerts from *vars* sources does not significantly impact our coverage of CV evolution because most CVs are very rapidly evolving and have fewer than $N_{\max} = 100$ alerts, so they typically receive no thinning at all.

Sources from all queries in the train split get $N_{\max}$ alert thinning to avoid the overrepresentation problem. In the validation and test splits, however, only sources from the *vars* query get alert thinning. The other sources are left unthinned because (i) overrepresentation is not an issue, and (ii) we need all alerts from these sources to most accurately measure completeness and purity of the `BTSbot` predictions. We keep alert thinning in place for the *vars* sources in the validation and test splits because the very old alerts from these long-lived AGN are unlike what appears in present-day scanning.

Figure 4 shows the number of sources and alerts in each split, and the query that they originated from after all cuts and thinning are applied. Our two classes are moderately imbalanced: ∼35% BTS alerts to ∼65% not-BTS alerts. We discuss techniques for mitigating the effects of class imbalance in Section 3. In total, our production training set comprises 608,943 total alerts from 19,558 total sources totaling >60 GB. Despite `BTSbot`'s relatively narrow domain, this is significantly more alerts than other models with similar architectures, such as the ALeRCE stamp classifier (∼52,000; Carrasco-Davis et al. 2021) and the Alert-Classifying Artificial Intelligence (ACAI) models (∼200,000; Duev & van der Walt 2021).

3. `BTSbot` Scope, Architecture, and Training

Our MM-CNN, `BTSbot`, automates source identification for BTS by assigning each ZTF alert packet a bright transient score. Figure 5 shows how input is fed into `BTSbot` and how information is combined to produce the output score; `BTSbot` contains three main components. (1) The convolutional branch processes the science, reference, and difference cutouts as a three-channel image through an architecture similar to what Simonyan & Zisserman (2014) propose. (2) The metadata branch processes the 25 extracted features (see Table 4) through a batch normalization layer and two dense layers. (3) The combined section concatenates the output of the two branches and passes it through two more dense layers, the second of which produces the prediction using a softmax activation function. The output is a unit-interval score where higher scores represent increased confidence that the source in the input alert packet is, or will become, a bright extragalactic transient. The parameters for each layer are shown in Table 2. Other than the final layer, activation functions are all the rectified linear unit (Nair & Hinton 2010), and all convolutional layers have the symmetric unit stride and same padding.

The choice of an MM-CNN is motivated by the fact that the images and the extracted features provide complementary information for performing our task. For example, the extracted feature `distpsnr1` represents the angular distance to the PS1 cataloged source nearest to this ZTF source, and `sgscore1` represents the star–galaxy score (Tachibana & Miller 2018) of this PS1 source. While new transients are not present in PS1, the host galaxies of SNe often are. Thus, alerts from SNe tend to have moderate `distpsnr1` and small `sgscore1` values, indicating a galaxy projected nearby to the source. Most AGN and some CVs that pass the BTS alert filter are cataloged in PS1 and thus have `distpsnr1` very near to zero. The images also provide important information following a similar heuristic. Bright SNe tend to be associated with prominent (bright with large angular size) off-center extended sources, their host galaxies; faint SNe tend to have less prominent host galaxies because they tend to be farther away; AGN will appear as exactly centered extended sources; CVs will often appear surrounded by many bright point sources because they tend to occur near the galactic plane around many other stars. Carrasco-Davis et al. (2021) give more detailed descriptions of how AGN, SNe, and VarStars can be distinguished using ZTF image cutouts and metadata. An MM-CNN is able to pool information from all input types and consider them together when making a prediction. We also experiment with unimodal alternatives to `BTSbot` in Appendix C.

Given the scope and architecture of `BTSbot`, we encounter a number of challenges. First, we are requiring `BTSbot` to learn multiple complex separations. `BTSbot` must learn to separate SNe from other sources without using redshift information because it is not always known a priori. It must also learn to identify bright SNe with limited time series information irrespective of the SN's current phase. Early in its rise or late in its fade, a bright SN can appear very similar to a

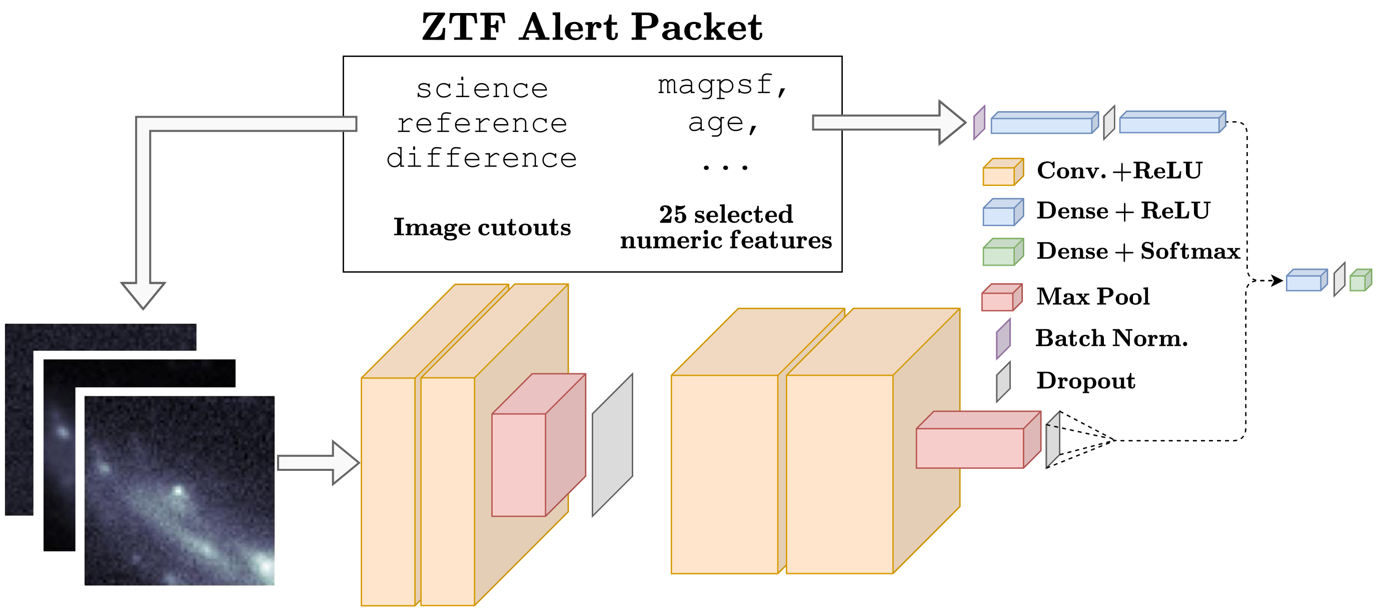


Figure 5. Diagram of our multimodal convolutional neural network, `BTSbot`, performing bright extragalactic transient/not bright extragalactic transient binary classification on ZTF alert packets. Image input is processed through the convolutional branch and then flattened to a 1D vector and concatenated with the output of dense layers in the metadata branch. After another dense layer, a single-neuron layer produces the final prediction: a unit-interval bright transient score.

Table 2
`BTSbot` Layer Configurations

Layer Type	Layer Parameters	Hyperparameter Search Range
	Convolutional Branch	
2D conv.	32 filters, 5 × 5 kernel	8–128 filters[a]
2D conv.	32 filters, 5 × 5 kernel	[3, 5, 7] kernel size[a]
Max pool	2 × 2 kernel	…
Dropout	0.50	0.1–0.8
2D conv.	64 filters, 5 × 5 kernel	8–128 filters[a]
2D conv.	64 filters, 5 × 5 kernel	[3, 5, 7] kernel size[a]
Max pool	4 × 4 kernel	…
Dropout	0.55	0.1–0.8
	Metadata Branch	
Batch norm.	…	…
Dense	128 units	32–256 units
Dropout	0.25	0.1–0.8
Dense	128 units	32–256 units
	Combined Section	
Dense	8 units	8–128 units
Dropout	0.20	0.1–0.8
Dense	1 unit	…

Notes. Images and metadata are passed through their respective branches, and the output of either is concatenated and sent to the combined branch. Dropout and batch normalization (batch norm.) layers are included to regularize.
[a] All 2D convolutional (conv.) layers have the same search range for filter counts and kernel size.

near-peak dim SN. Additionally, `BTSbot` does not have explicit information stating the BTS threshold is 18.5 mag. Rather, it must learn the position of this threshold drawn through the continuous distribution of SNe peak magnitudes. Lastly, `BTSbot` does not have direct access to full light-curve information, i.e., information on every previous individual detection. Instead, it has access to the basic custom metadata features we compute from the full light curve. We omit full light-curve information, in part, because there is no established method in the literature for representing partial light curves of the wide variety our model encounters in a way that is fit for input into an NN. There has been a great deal of work to accomplish this for SNe alone (e.g., Villar et al. 2020), but these methods are not applicable to all the types of sources that `BTSbot` encounters. This choice supports the possibility of tuning `BTSbot` to identify extragalactic transients very rapidly, potentially on their first detection, which Section 5.3 explores.

3.1. Training and Hyperparameter Optimization

`BTSbot` is implemented with `TensorFlow` (Abadi et al. 2016) and the Keras application programming interface (API).[23] We adopt the Adam optimizer (Kingma & Ba 2014) and the binary cross-entropy loss function. In addition to thinning a source's alerts by $N_{\max}$ (see Section 2), we make use of a number of training techniques to mitigate overfitting. We employ data augmentation, which executes random rotations of 0°, 90°, 180°, and 270° and random horizontal and vertical flipping on the image cutouts. These also help ensure that `BTSbot` is invariant to these transformations. We weight contributions to the loss function by the inverse of the relative size of the input alert's class (i.e., misclassifications of BTS alerts contribute more loss than that of not-BTS alerts). This size is computed by the count of alerts per class in the train split, but we also experiment with computing these weights based on the number of sources per class. The learning rate α decreases after 20 epochs without improved validation loss, and training terminates after 75 epochs without an improved validation loss.

We utilize the Weights and Biases platform[24] to perform multiple extensive Bayesian hyperparameter sweeps. A Bayesian hyperparameter sweep guides searching through a large

[23] https://keras.io
[24] https://wandb.ai/site/

Table 3
BTSbot Hyperparameters

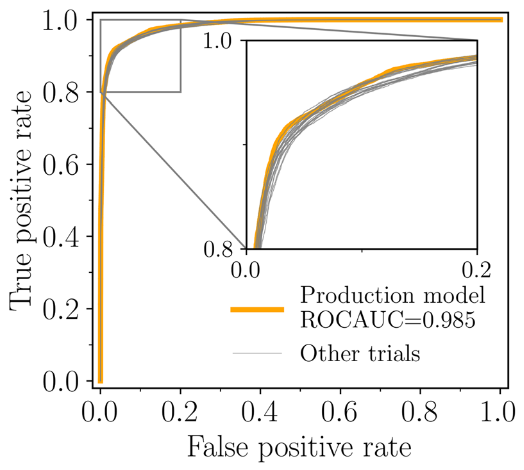
Parameter Name	Optimized Value	Hyperparameter Search Range
Batch size	64	8–64
Adam β_1	0.99	0.81–0.999
Adam β_2	0.99	0.9–0.9999
Learning rate (α)	10^{-4}	10^{-2}–5×10^{-6}
α decrease factor	0.4	0.25–0.75
$\alpha_{\min}$	5×10^{-10}	10^{-10}–10^{-5}
$N_{\max}$	100	1–∞

Table 4
BTSbot Metadata Features

Feature Name	Definition (unit)
Alert Packet Metadata	
sgscore{1,2}	Star/galaxy score of nearest two PS1 sources
distpsnr{1,2}	Distance to nearest two PS1 sources [arcsec]
fwhm	Full width half max [pixels]
magpsf	Magnitude of PSF-fit photometry [mag]
sigmapsf	1σ uncertainty in magpsf [mag]
chipsf	Reduced χ^2 of PSF-fit
ra	R.A. of source [deg]
dec	Decl. of source [deg]
diffmaglim	5σ magnitude detection threshold [mag]
ndethist	Number of previous detections of source
nmtchps	No. of PS1 crossmatches within 30″
drb	Deep learning-based real/bogus score
ncovhist	No. of times source on a field and read channel
chinr	χ parameter of nearest source in reference
sharpnr	Sharp parameter of nearest source in reference
scorr	Peak-pixel S/N in detection image
sky	Local sky background estimate [data number]
Custom Metadata	
days_since_peak	Time since brightest alert [days]
days_to_peak	Time from first to brightest alert [days]
age	days_since_peak + days_to_peak
peakmag_so_far	Source's minimum magpsf thus far [mag]
maxmag_so_far	Source's maximum magpsf thus far [mag]
nnondet[a]	ncovhist - ndethist

Notes. The 25 metadata features passed into BTSbot's metadata branch. Full definitions of alert packet features can be found at https://zwickytransientfacility.github.io/ztf-avro-alert/schema.html. Acronyms: point-spread function (PSF), signal-to-noise ratio (S/N).
[a] Adopted from Carrasco-Davis et al. (2021).

grid of hyperparameters intelligently by leveraging correlations between previous hyperparameter inputs and the corresponding trained model's performance in a specified metric. This is expected to be more efficient than traditional grid search sweeps because, for example, the Bayesian sweep can learn to avoid certain regions of the hyperparameter space that frequently yield poor performing models. We find the optimal configuration of hyperparameters for BTSbot layers (see Table 2), other numeric values (see Table 3), and the selection of metadata features (see Table 4). Tables 2 and 3 also include the range of values searched over for each hyperparameter in our sweeps. The search over which alert packet metadata features to provide to BTSbot included many other features that we found to not improve performance: sgscore3, distpsnr3, fid, maggaia, neargaia, magdiff, magap, sigmaap, magapbig, sigmaapbig, magnr, ssnrms, dsnrms, seeratio, nneg, magzpsci, jdstarthist, and classtar.



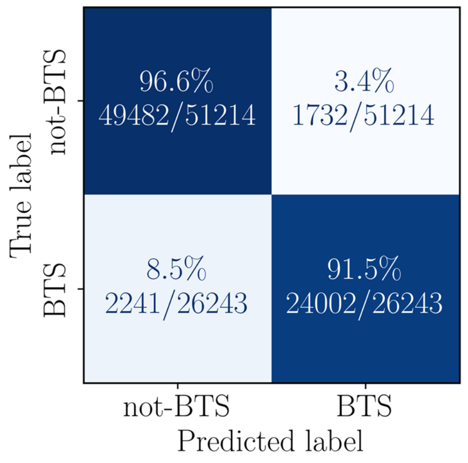


Figure 6. ROC curve (top) and confusion matrix (bottom) of the BTSbot production model and 19 other trials with different initializations. Top: the production BTSbot model (orange curve) and 19 other trials (gray curves) yield very similar ROC curves and ROCAUCs, indicating that BTSbot's excellent performance is robust to different initializations. Bottom: The production model's confusion matrix makes clear that BTSbot has greater TN rate (upper left quadrant) than TP rate (lower right quadrant). We find that models that trade off TP rate in favor of TN rate tend to improve in other key performance metrics.

4. BTSbot Performance and Comparison to Human Scanners

4.1. Performance on the Test Split

To characterize BTSbot's generalizable performance, we present performance metrics on test split data. We train 20 trials of BTSbot with identical optimal hyperparameters and different random initializations of the model's learnable parameters. The trial that yields the best test split performance is chosen as the production model. Performance metrics are reported for the chosen production model and the median of the metric for the 20 trials with uncertainties representing the metric's 1σ bounds.

The final BTSbot models yield a test accuracy of $94.1\% \pm 0.28$ and 94.9% for the production model. The upper panel of Figure 6 shows BTSbot's receiver operating characteristic (ROC) curves. ROC curves visualize the balance of true positives (TPs) and false positives (FPs) at various classification thresholds, and the area under the ROC curve (ROCAUC) is frequently used as a summary statistic for the performance of a classifier. A perfect classifier's ROC curve has ROCAUC equal to unity while ROCAUC = 0.5 corresponds to random guessing. The final `BTSbot` models yield ROCAUC = 0.984 ± 0.001 and ROCAUC = 0.985 for the production model. The results of the production model and the 19 other trials are extremely similar, indicating stability in `BTSbot`'s performance over different initializations. The lower panel of Figure 6 shows the production model's confusion matrix. A binary classifier's confusion matrix visualizes the frequencies of the four classification outcomes: true negative (TN), FP, false negative (FN), and TP. The production model shows higher accuracy on not-BTS alerts (TN rate) than on BTS alerts (TP rate). We could attempt to train `BTSbot` to better balance the TP and TN rates, for example by adjusting the class weights, but we find that models that favor TN rate perform better overall.

While we fare excellently in traditional ML performance metrics, they are not necessarily representative of `BTSbot`'s real-world performance. Because `BTSbot` produces scores on alert packets (alert-based classification) but sources must be chosen for follow-up (source-based classification), we must define a mapping from a sequence of alert predictions to a source prediction. We call these mappings "policies," which we define to be analogous to the criteria BTS scanners use when deciding whether or not to save a source and request a spectrum of it. By simulating our policies on BTS candidates, we can compute performance metrics that realistically represent how `BTSbot` performs as a scanner. Further, we can compare the resulting figures against human scanners to contextualize `BTSbot`'s performance.

We define two policies that closely emulate their human scanning analogs: `bts_p1` and `bts_p2`. The policy `bts_p1` requires that a source have at least two alerts with high ($\geqslant$0.5) bright transient score and `magpsf` $\leqslant 19$ mag before being saved and having an SEDM trigger sent at priority 1. The policy `bts_p2` requires that a source meet `bts_p1` as well as having at least one alert with `magpsf` $\leqslant 18.5$ mag before a trigger being sent with priority 2. Priority is a parameter of the requests sent to SEDM, where larger values indicate the request is more urgent. Higher priority requests are typically fulfilled before lower priority requests, although other factors, e.g., observability, are also taken into account. Most follow-up requests sent by BTS scanners are with priority 1 or 2, and priorities 2 and greater are typically reserved for sources that have already attained $m_{\rm peak} \leqslant 18.5$ mag.

We quantify the performance of these policies with completeness, purity, $\Delta t_{\rm save}$, and $\Delta t_{\rm trigger}$. Completeness (or "recall") is the fraction of bright transients that are correctly classified by the policy: TP/(TP + FN). The completeness of `bts_p1` and `bts_p2` are identical, so we report a single value for both policies. Purity (or "precision") is the fraction of predicted bright transients that are actually bright transients: TP/(TP + FP). $\Delta t_{\rm save/trigger}$, is the difference between the Julian date (JD) at which `BTSbot` saved and triggered on a source with the JD that scanners did the same. The JDs for scanners are queried from `Fritz` for all sources in the *trues* set (see Section 2). Sources that were saved before 2021 January 1 are removed from this analysis because many of them were scanned using the GROWTH Marshal (Kasliwal et al. 2019), so the save and trigger JDs available on Fritz are unreliable. For saving, we use the JD at which a scanner added a source to the BTS catalog on `Fritz`. For triggering, we only consider sources that had an SEDM integral field unit (IFU) request sent before their first spectrum was uploaded to `Fritz`. The JD used for comparison is the JD at which the first IFU follow-up request was created. This restriction is applied because the time at which scanners decided to trigger is not available for the other facilities BTS uses for classification. `BTSbot`'s JD for saving and triggering on a source is the JD associated with the alert that first made the source satisfy `bts_p1`. The `BTSbot` JDs correspond to either policy, but we select `bts_p1` as it makes for a more direct comparison to scanners.

When computing completeness and purity for these policies, we make two additional minor cuts. First, 70 sources in the test split also appear in *junk* (see Section 2). This catalog is a list of sources that are unambiguously not bright transients, which frequently pass the BTS alert filter. When scanning, they are typically hidden and not considered for saving or triggering; they are excluded from completeness and purity calculations. We also identify 59 sources that have only a single alert remaining after the cleaning cuts (see Section 2). These sources will never pass either policy, so they are also excluded from completeness and purity calculations.

We use estimates of the scanners' completeness, purity, and speed as benchmarks against which to judge `BTSbot`'s performance. We compute a lower-limit on the scanners' saving completeness of bright transients from values presented in Table 1 and Section 3 of Perley et al. (2020), which give 99.6% completeness. It is not straightforward to compute the BTS scanners' purity for saving or triggering on bright transients. Scanners will often intentionally act on transients that they know will not attain $m_{\rm peak} \leqslant 18.5$ mag,[25] so straightforward estimates of the scanners' bright transient purity would underestimate their ability to select only bright transients. Instead, we estimate their saving and triggering purity for selecting any extragalactic transient and rejecting other sources. We limit this analysis to saves and triggers performed between $2460175.5 < {\rm JD} < 2460216.5$ (see Section 4.2 for explanation of JD bounds). During this time, 266 sources were saved by scanners to the primary internal BTS catalog, only four of which were not extragalactic transients: 98.5% scanner saving purity. Similarly, scanners sent SEDM IFU for 327 unique sources of which 11 were nonextragalactic transients: 96.7% scanner triggering purity. Further, scanners saving and triggering on transients with $m_{\rm peak} > 18.5$ mag complicates $\Delta t_{\rm save/trigger}$ estimates. Scanners will save and trigger on transients slightly faster than otherwise because they need not wait for a transient to unambiguously demonstrate that it will soon have $m_{\rm peak} \leqslant 18.5$ mag. In addition, BTS scanners are occasionally aided in identifying bright transients by TNS reports from scanners working with other surveys or other ZTF alert brokers. For these reasons, $\Delta t_{\rm save/trigger}$ comparisons will not be perfectly direct, but we still adopt them as a basic benchmark for `BTSbot`'s speed.

The left panel of Figure 7 shows `BTSbot`'s completeness and purity under our policies as a function of peak magnitude. The completeness curve is exactly 100% in all peak magnitude bins, giving perfect overall completeness. This compares well

[25] These sources, while not "bright" transients, are still of interest to BTS and the individuals within BTS.

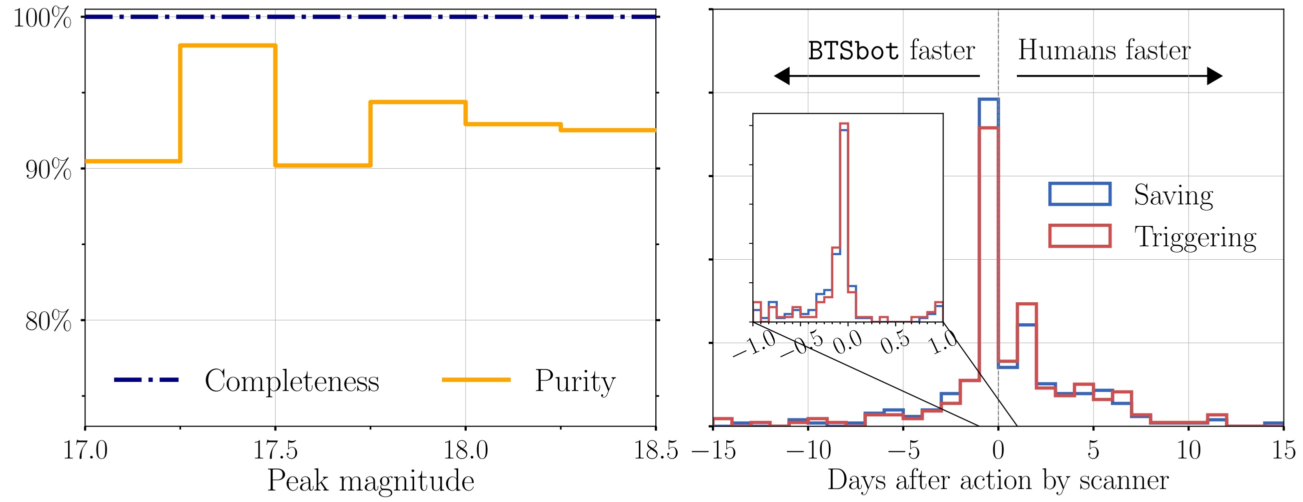


Figure 7. Completeness and purity of `BTSbot` actions (left) and speed comparison with human scanners (right) for sources in our test split. Left: The completeness curve (dashed–dotted navy) is 100% in all bins. `BTSbot`'s perfect completeness is conducive to BTS's science efforts, which require a highly complete, unbiased sample. The small variations in the purity curve (solid orange) are due to small number statistics. The overall purity of `bts_p1` and `bts_p2` are 84.6% and 93.0% respectively. `bts_p1` has extra contamination from SNe with $m_{\rm peak}$ slightly greater than 18.5 mag, sources that are acceptable targets for spectroscopic observation. Right: Histograms comparing `BTSbot`'s speed in saving (blue) and triggering (red) with that of scanners. Both distributions peak very near to zero indicating that `BTSbot` acts as quickly as human scanners on new bright transients.

to the BTS scanners' saving completeness: 99.6%. Very high completeness is essential for maintaining the quality of the BTS sample and ensuring that `BTSbot` does not inject any significant selection biases into the BTS sample. The purity curve is >90% in all peak magnitude bins. Most bins have <10 FPs, so the variations between bins is likely due to small number statistics. The overall purity for `bts_p2` is 93.0%, and the overall purity for `bts_p1` is 84.6%. The `bts_p1` purity is not aligned with the purity curve because `bts_p1` also considers a large number of sources with $m_{\rm peak} > 18.5$.[26] Contamination from dim ($m_{\rm peak} > 18.5$) transients is the reason why the `bts_p1` purity is much less than the `bts_p2` purity; they make up 53 of the 54 FPs unique to `bts_p1`. These 53 transients have median $m_{\rm peak}$ of 18.57 mag, and many received spectroscopic follow-up requests by BTS. They are still of interest to members of BTS, so they represent a reasonable use of follow-up resources. `BTSbot`'s purity falls slightly (∼2%–6%) short of the scanners' (98.5% saving, 95.6% triggering). Likely at the cost of completeness or speed, alternative policies could be designed to more conservatively allocate spectroscopic resources, increasing `BTSbot`'s purity to compare favorably with scanners. `BTSbot`'s behavior is dependent on `braai` and `sgscore` because they contribute to determining what sources pass the BTS alert filter: the only sources that `BTSbot` trains and triggers on.

The right panel of Figure 7 shows histograms comparing the time at which `BTSbot` and the human scanners saved or triggered on a source. Negative values indicate that `BTSbot` was faster, and positive values indicate that the scanners were faster. Both histograms peak sharply at 0 days, suggesting that scanners and `BTSbot` act on new transients at the same time. The median of $\Delta t_{\rm save}$ and $\Delta t_{\rm trigger}$ are –0.0381 and –0.0147 days respectively; `BTSbot` acts marginally quicker than scanners. Much of this performance is likely due to `BTSbot`'s decisions being made immediately as new alerts filter through the ZTF and BTS pipelines, although BTS does benefit from consistent real-time scanning thanks to members in European time zones.

The tails of this distribution include six (11) sources that are saved by `BTSbot` a week or more before (after) scanners did. Nuclear and hostless SNe make up most of the cases where `BTSbot` was faster. This suggests `BTSbot` is less hesitant to claim these challenging sources to be bright transients. Most of the cases where `BTSbot` is slower than scanners are slowly evolving SNe with a history of detections down to ∼20 mag. These suggest that scanners can better use the evolution prior to reaching 19 mag to identify transients, and `BTSbot` needs brighter detections to identify sources. Late identifications of these sorts are unlikely an issue because `BTSbot` still consistently identifies these sources before or near peak. Sources with poor light-curve coverage, especially around 19 mag, cause large $\Delta t_{\rm save}$ and $\Delta t_{\rm trigger}$ and appear in either of these groups. Overall, `BTSbot` fares very well in time to save and trigger when compared to scanners.

4.1.1. Analysis of Misclassifications in the Test Split

Tracking and categorizing misclassifications is a key part of the development of ML models. Misclassifications are particularly important to understand in the case of `BTSbot` because mistakes could introduce biases into the BTS sample and waste valuable spectroscopic resources.

On test split data, `bts_p1` selects 92 FPs and zero FNs. The majority of the FPs (53/92) are real dim transients. These are sources outside of `BTSbot`'s positive class but are still of interest to BTS, so they are acceptable FPs. Nearly all of the remaining FPs are CVs, AGN, or QSOs and are shared with `bts_p2`. The center panel of Figure 8 presents an instance of a typical `bts_p1` FP. Alerts are classified as not belonging to a bright transient until the source nears the 18.5 mag threshold, at which point a small number of high-scoring alerts cause the source to pass `bts_p1`. The right panel shows an FP CV misclassified by both `bts_p1` and `bts_p2`. It shows high- and low-scoring alerts interspersed with each other, although some other CV FPs begin receiving exclusively low scores

[26] Completeness and purity are not shown for $m_{\rm peak} > 18.5$ because they are undefined and uniformly zero respectively.

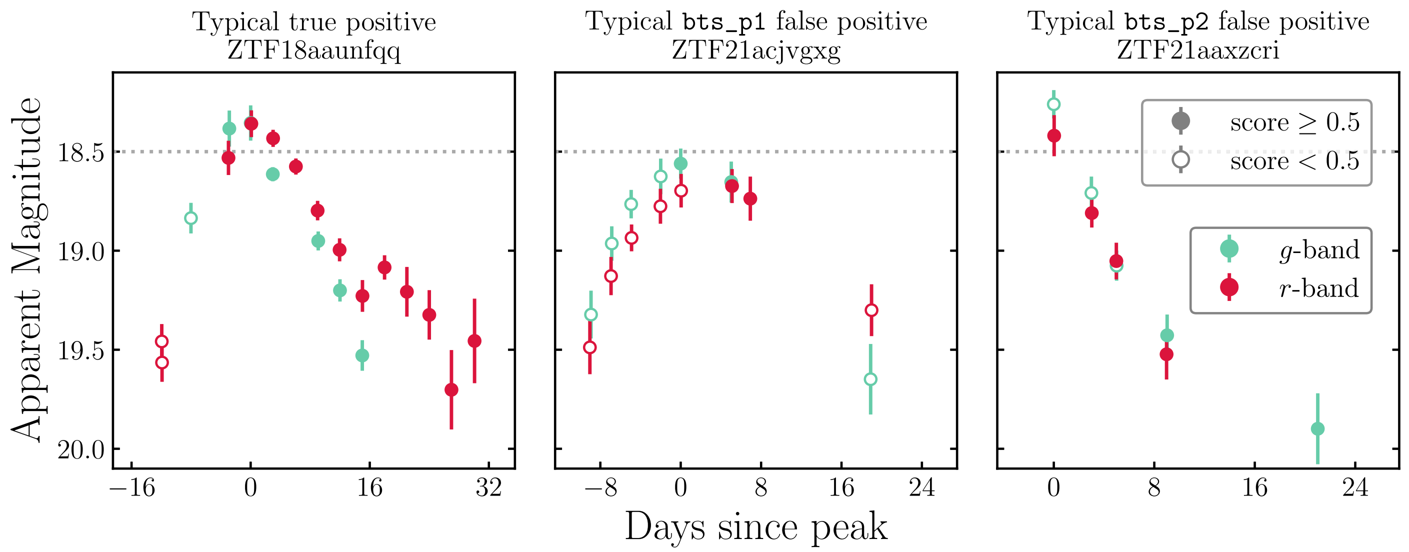


Figure 8. Light curves of three sources depicting typical evolution of `BTSbot` scores. Teal (g band) and red (r band) points indicate detections, and filled and open circles represent alerts that received score $\geqslant$0.5 and <0.5 respectively. Left: TPs may have low-scoring alerts while still dim, but scores increase once they near the 18.5 mag threshold (dotted gray line). After fading well below the peak magnitude, scores remain high, in part, due to information provided by custom metadata features. Center: almost all `bts_p1` FPs are dim ($m_{\text{peak}} > 18.5$ mag) transients whose alerts receive high scores when near the BTS threshold. Right: many `bts_p2` FPs are CVs, which could be better rejected by increasing the score threshold to 0.8.

once the CV has faded well beyond peak. The completeness is perfect for `bts_p1` and `bts_p2`, so they have zero FNs.

Overall, the non-SN misclassifications are dominated by a relatively small number of CVs, AGN, and QSOs. Improved policies may be able to improve the rejection of these frequent FPs by, e.g., better leveraging existing information in external catalogs. The majority of the total misclassifications (dim SNe selected by `bts_p1`) are not problematic for BTS and represent an appropriate allocation of spectroscopic time.

4.2. Performance on Very Recent BTS Candidates

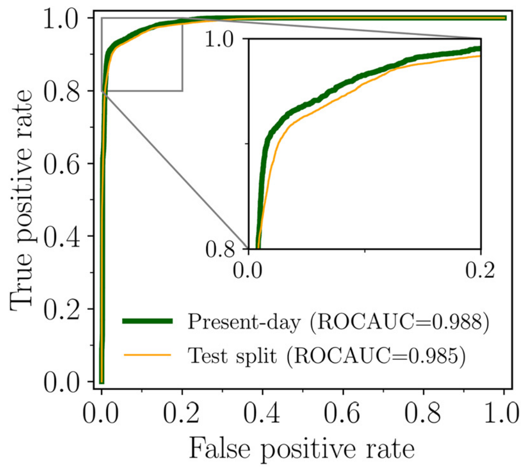
Performance diagnostics computed on test split data tend to be robust and representative of real-world performance, but, in some cases, can have associated biases. Our test split includes many alerts that are years old, and a subtle data shift (caused by, e.g., maintenance to the camera or the optics) may have occurred since then. To characterize `BTSbot`'s present-day performance, we conduct an additional analysis using alerts from very recent BTS candidates.

We perform this analysis in two parts: (i) we present alert-based performance metrics on alerts that recently passed the BTS alert filter and our cleaning cuts (see Section 2); (ii) we also present policy-based performance metrics on sources that recently passed `bts_p1` or recently peaked. The date boundaries for these analyses are determined by the final date our training data was queried (2023 August 19) and the date `BTSbot` went into production (2023 September 29): $2460175.5 < \text{JD} < 2460216.5$. This cut on JD minimizes the bias in this analysis from transient alerts that `BTSbot` trained on and instances where `BTSbot`'s actions, which are visible to scanners, influenced scanners' decisions. Some very long-lived sources, like certain AGN, do have alerts in our training data and in this cleaned present-day sample. We do not remove these sources because they are encountered by `BTSbot` in production, and thus should be accounted for in this analysis.

We begin this analysis by applying the cleaning cuts described in Section 2 on the public alerts that passed the BTS filter with $2460175.5 < \text{JD} < 2460216.5$. After cuts, this sample totals 4031 alerts from 251 bright transients and 15,159 alerts from 1652 non-bright transients. Figure 9 shows that the production `BTSbot` model yields 96.2% accuracy and ROCAUC = 0.988 on the present-day sample. TP rate and TN rate are 91.9% and 97.4%, respectively. The resulting performance is very similar to the metrics computed from test split data in Section 4.1 and shown in Figure 7. Here, we observe marginally higher performance across all alert-based metrics than in the test split results. These variations are most likely due to the relatively small size of the present-day sample and are not indicative of a data shift affecting `BTSbot`'s performance.

As in Section 4.1, any source in *junk* or having only one alert after cleaning is removed when computing policy completeness and purity. In Section 2, the *vars* query received alert thinning down to $N_{\max} = 100$ alerts per source. We emulate this by thinning sources classified on `Fritz` or manually identified as an AGN, CV, or QSO down to their latest $N_{\max} = 100$ alerts. We run all public alerts passing our Section 2 cleaning cuts from all sources through both policies and select only the sources that satisfy `bts_p1` or reach their peak magnitude between $2460175.5 < \text{JD} < 2460216.5$. This representatively simulates the actions `BTSbot` would have taken during this time period should it have been fully operational.

Similar to Figure 7, Figure 10 shows the completeness, purity, and speed comparison of the production `BTSbot` model on the present-day BTS sample. Both the completeness and purity curve, shown in the left panel, are at or near 100% in most peak magnitude bins. The overall completeness is 98.7%. This remains near perfect and supports `BTSbot`'s ability to scan without imposing significant selection biases into the BTS sample. Variations in either curve are due to small number statistics; there are 0–3 FPs and FNs in each bin shown. The overall `bts_p1` purity is 81.0%, and the overall `bts_p2` purity is 92.0%. Similar to the test split results, the sources selected by `bts_p1` but not `bts_p2` are dominated by SNe with m_{peak} slightly dimmer than 18.5 mag (19/22 sources): unproblematic FPs because they are typically triggered on by



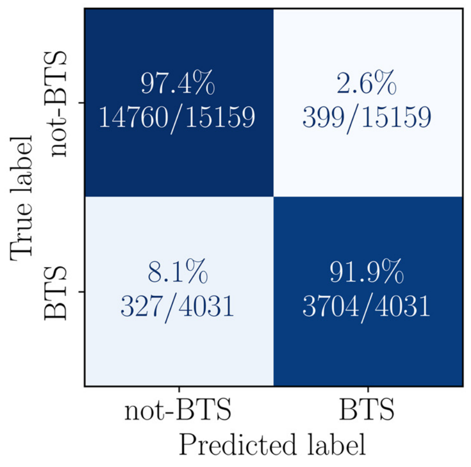


Figure 9. Same as Figure 6 for a sample of alerts from very recent BTS candidates. The results of the ROC (present-day, thick green; test split, narrow orange), the ROCAUC, and the confusion matrix are all very similar to their test split analogs. All metrics are marginally improved for the present-day sample but not significantly so. These suggest that no data shift has occurred that significantly decreases `BTSbot`'s performance.

BTS. The completeness and purity estimates of the present-day sample are very similar to their test split analogs, $\leqslant \pm 4\%$ difference in all three metrics, further suggesting that no data shift has affected `BTSbot`'s performance.

The right panel of Figure 10 compares `BTSbot`'s speed to act with that of the human scanners on our present-day sample. The medians of $\Delta t_{\rm save}$ and $\Delta t_{\rm trigger}$ are larger than their test split counterparts: 0.0525 and 1.01 days respectively. We attribute this increase to two main factors. (i) We have excluded ZTF partnership data from the present-day sample, but scanners typically view data from both the ZTF public and partnership observations when scanning. This frequently provides them with additional detections for BTS candidates, which `BTSbot` is blinded from, aiding in more quickly identifying bright transients. (ii) The present-day sample is very small, so the median is volatile to changes. There are just 65 sources with $\Delta t_{\rm save}$ values in the present-day sample, while there were >250 sources for the test split. The shapes of the Δt distributions, however, are consistent between the present-day and test split analysis. Together, these suggest that the differences observed are not due to a data shift but rather to differences in the sample characteristics and in how the experiments were conducted.

With a sample of BTS candidates contiguous in time, we can now easily compute the median number of saves and triggers performed by `BTSbot` per night. Over the 41 nights in the present-day sample, 184 sources satisfied `bts_p1`. This is a median of ~4.5 sources per night, which would have been saved and sent to SEDM.

4.2.1. Analysis of Misclassifications in the Present-day Sample

Analogously to Section 4.1.1, we investigate the types of sources that `BTSbot` misclassifies in the present-day sample.

Our policies select 35 FPs and two FNs. Similarly to the test split results, the majority of FPs (19/35) are real dim transients. The other FPs are again dominated by CVs and AGN but do include other sources like asteroids. One of the FNs (ZTF23aaxtplp) has an extremely bright host galaxy and is projected very near to its nucleus. The other FN (ZTF23abeuope) shows two bright stars overlapping the host galaxy and projected very near to the SN. It is not certain that these properties caused these sources to receive low scores, but they are clearly very uncommon. Section 5.1 and Appendix B describe how, in production, `BTSbot` and humans scan in parallel to allow for significant automation while preventing bright transients of these sorts from being missed.

As is the case for the test split, the majority of the misclassifications in the present-day sample, dim SNe, have little negative consequence for BTS.

5. Discussion

5.1. `BTSbot` Real-time Operations

As of 2023 October, `BTSbot` has been fully integrated into ZTF and BTS operations,[27] allowing `BTSbot` to autonomously send spectroscopic triggers for new SNe. In production, `BTSbot` and humans scan the real-time public alert stream separately, allowing both parties a chance to save any source passing the BTS alert filter irrespective of the other party's actions. Technical details of `BTSbot`'s integration and this parallel scanning are presented in Appendix B.

Between 2023 December 1 and 2024 May 1, `BTSbot` saved 609 sources to an internal BTS catalog, and 96.1% of these were confirmed as extragalactic transients. SN 2023tyk (ZTF23abhvlji) is a Type Ia SN that was identified and triggered on by `BTSbot`. The data collected by SEDM were then reduced by `pySEDM` (Rigault et al. 2019), the spectrum was classified as belonging to a Type Ia SN by `SNIascore` (Fremling et al. 2021), and the classification was automatically reported to TNS. As detailed by Rehemtulla et al. (2023a), SN 2023tyk is the first transient to be fully automatically detected, identified, spectroscopically classified, and publicly reported. As of 2024 May, dozens more Type Ia SNe have been both triggered on by `BTSbot` and spectroscopically classified by `SNIascore`.

Operating `BTSbot` and `SNIascore` alongside each other allows for the full-automation of a significant amount of day-to-day tasks in BTS. About 70% of bright transients found by BTS are Type Ia SNe (Fremling et al. 2020), nearly all of

[27] An older version of `BTSbot` (v1.0) was running in production during 2023 October and November. The production model presented here (v1.0.1; Rehemtulla et al. 2024), deployed in 2023 December, corrects for the sources with $m_{\rm peak} \leqslant 18.5$ in the *dims* query (see Section 2). Both versions of the model yield nearly identical performance, but v1.0.1 is accompanied by a policy that prioritizes purity slightly more.

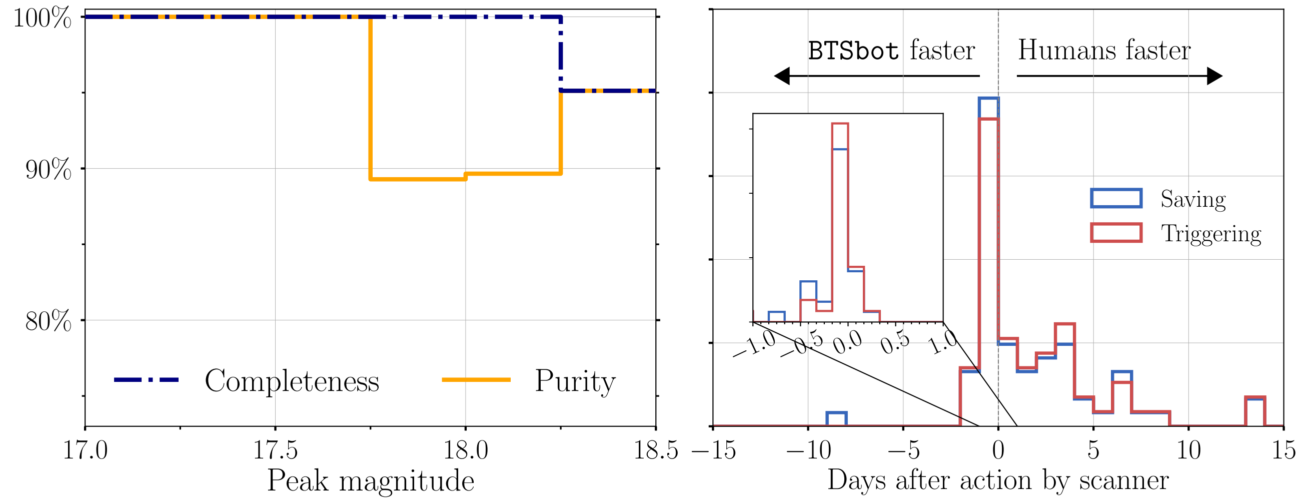


Figure 10. Same as Figure 7 for a sample of very recent BTS candidates. Left: The completeness (dashed–dotted navy) and purity (solid orange) curves are perfect in most bins. The overall purity of `bts_p1` and `bts_p2` are 81.0% and 92.0% respectively. Contamination unique to `bts_p1` is almost entirely real SNe with m_{peak} slightly greater than 18.5, acceptable false positives. The relatively low purity in two bins is due to small number statistics. Right: Both distributions still peak very near to zero. Likely due to differences in the cuts creating the input samples, Δt_{save} (0.0525 days) and $\Delta t_{\text{trigger}}$ (1.01 days) are larger than test split equivalents.

which we expect to be identified by `BTSbot` and 80%–90% of which will be classified and reported to TNS by `SNIascore`. If no human scanning took place, this BTS workflow could fully automatically handle more than half of the bright transients in consideration by BTS. In practice, there are additional tasks that remain to be automated or made more efficient. Namely, scanning also involves the retrieval of host galaxy redshifts from the NASA Extragalactic Database (NED) and classifications from TNS. We expect that much of these processes can be automated, and planning for doing so is underway. While a comprehensive transition to automated scanning for BTS is not the target, these tools mitigate the dependence of BTS operations on the natural fluctuations of scanners' lives. Further, a reallocation of some human effort invested in BTS is possible when many tasks are automated. BTS experts are able to spend more time, e.g., analyzing bulk properties of SN samples or searching for rare or very young events.

5.2. Comparison with Similar Models

The ALeRCE[28] (Förster et al. 2021) real-time stamp classifier (Carrasco-Davis et al. 2021) and the ACAI (Duev & van der Walt 2021) framework are other image- and extracted feature-based MM-CNNs that perform classification on ZTF alert packets. `BTSbot` is very similar to these models in architecture (indeed, some aspects of `BTSbot` are inspired by them), but they are quite different in application.

The real-time stamp classifier predicts which one of five high-level classes (SN, AGN, VarStar, asteroid, bogus) the source in a ZTF alert belongs to. This is done, primarily, with the goal of automatic and rapid identification of SNe. To this end, the stamp classifier is trained exclusively on the first alert packet from a given source. While rapid identification of SNe is an interest of ours (see Section 5.3), we instead train `BTSbot` to function on any alert packet from relevant ZTF sources. This grants `BTSbot` the extra utility of being able to identify SNe at any phase of their evolution, albeit with the additional complications associated with doing so.

[28] https://alerce.online/

The ACAI framework, a union of five independent binary classifiers, predicts which of five phenomenological features (hosted, orphan, nuclear, variable star, bogus) characterizes the source in a ZTF alert packet. These models are trained on any alert from any ZTF source and thus learn a much broader domain than `BTSbot`. Narrowing the input domain of our model reduces its broader utility relative to ACAI but unlocks greater performance for our particular task.

Unlike `BTSbot`, neither the stamp classifier nor ACAI learn class definitions that are sensitive to the source's brightness. `BTSbot` must learn to select just a subset of extragalactic transients, those with $m_{\text{peak}} \leqslant 18.5$ mag, and reject other extragalactic transients and all other sources. While `BTSbot` performs binary classification, effectively grouping non-bright transient classes, the stamp classifier performs five-class classification, and the ACAI models perform five binary-classifications. This requires these other models, the stamp classifier in particular, to learn more discriminatory information between its five classes. This is not necessary for `BTSbot` because our primary interest is automating BTS scanning, for which only bright transients are relevant. Thus, we can justify combining non-bright transients into a single class, simplifying `BTSbot`'s task.

5.3. An Adaptation of `BTSbot`: Automatic, Very Rapid Follow-up

The development of `BTSbot` and the design of our policies prioritized completeness of bright transients over other metrics like purity and speed. After operating in production for a few months, a different set of policies was implemented, which are focused on increasing purity to operate `BTSbot` with minimal intervention. These priorities are directed by the needs of BTS, but applications of a `BTSbot`-like model to other science efforts could prefer to prioritize other metrics.

`BTSbot`'s architecture is particularly well suited for the automated identification of very young transients. At early times (i.e., the night of the first detection), the source's light curve is uninformative because it is comprised of a very small number of data points. Instead, much of the information available on the new transient is embedded in associated images. In this regime, one would expect that an image-based model, like BTSbot, would have better access to constraining information than a purely light-curve based model.

We take SN 2023ixf as an example illustrative of the additional discovery potential of a model like BTSbot. SN 2023ixf was a Type II SN in Messier 101. Owing to its proximity, it provided a unique laboratory to study pre-SN mass-loss events of red supergiants, allowed the assembly of a comprehensive time-dependent description of the event, and much more (e.g., Bostroem et al. 2023; Hiramatsu et al. 2023; Jacobson-Galán et al. 2023; Qin et al. 2023; Zimmerman et al. 2024). SN 2023ixf was reported to TNS by Koichi Itagaki at 21:42 UTC on 2023 May 19 (Itagaki 2023). The earliest published spectrum was collected less than an hour later by Perley et al. (2023). About 14 hr before the first TNS report, SN 2023ixf was detected by ZTF, and, just minutes later, this alert packet was assigned a bright transient score of 0.840 by an early version of BTSbot.[29] A variant of BTSbot could be trained to isolate such sources. With an alert filter and policy suited for the search of young transients, this could have allowed for a more rapid identification and spectroscopic follow-up of SN 2023ixf. About half the observing night was remaining for SEDM at the time the first detection would have passed the autotriggering filters, enough time to obtain a spectrum if triggered at high enough priority. If the spectrum is collected near the end of observing that night at Palomar Observatory (∼12:00 UTC), this represents a ∼10 hr speed-up over the otherwise earliest spectrum obtained. In this example, BTSbot and the associated integration tools presented here allow the probing of early, rapidly evolving explosion physics typically unavailable to traditional triggering methods.

There are a number of challenges related to this adaptation of BTSbot. Namely, the BTS alert filter and the bts_p1 and bts_p2 policies presented here would likely be inappropriate or suboptimal following this new definition of the model's priorities. A thorough exploration of how to best assemble a training set for this goal would also be required.

The automatic spectroscopic follow-up of infant SNe in ZTF data has been successfully implemented before in AMPEL (Nordin et al. 2019). The target selection was driven by SNGuess (Miranda et al. 2022), a decision tree-based ML system for identifying SNe. Their automated triggering was designed to observe nearby infant SNe, which was successfully done for a number of sources.

BTSbot's source code and the production BTSbot model are publicly available on GitHub,[30] and the public training data are made available on Zenodo at doi:10.5281/zenodo.10839691 (Rehemtulla 2024). This repository includes all codes necessary for assembling BTSbot's training set, training the model, creating figures visualizing validation or test split performance, and more. It is written specifically with adaptability and flexibility in mind. Additional functionalities, e.g., training models with alternative architectures (see Appendix C), are embedded in the scripts with minimal added complexity and no repeated code. This is done to facilitate ease in recycling the BTSbot code-base for other applications. One could, for example, quickly explore solving a problem with a simple fully connected NN and later advance to an MM-CNN with powerful features like data augmentation and Weights and Biases hyperparameter sweeps integration already built-in. A workflow of this sort is promising given the rapid training times and near-BTSbot performance demonstrated by the NN in Appendix C. We encourage the use of this resource by the community.

[29] The early version of BTSbot referenced here (v0.3) is presented in Rehemtulla et al. (2023b).
[30] https://github.com/nabeelre/BTSbot

6. Summary

We have presented BTSbot, a new MM-CNN to automate scanning for the ZTF BTS. BTSbot uses ZTF image cutouts and metadata to produce a bright transient score for an individual ZTF alert packet. It achieves ∼95% accuracy on input alerts and identified 100% of bright transients in our test split with 93% purity. The performance compares very closely to that of human scanners in terms of completeness, purity, and speed to act on new transients. BTSbot only falls slightly short of scanners in terms of the purity of the bright transient sample it produces: 93% versus ∼97%. We also perform an additional analysis with very recent BTS candidates to demonstrate that BTSbot is not impacted by a significant data shift.

BTSbot has been fully integrated into Kowalski and Fritz, ZTF's first-party alert broker and marshal, and now automatically sends spectroscopic follow-up requests for the new bright transients it identifies. BTSbot joins a family of automation tools in the BTS workflow (braai, sgscore, pySEDM, and SNIascore), which aid in running BTS efficiently. These models, coordinated by Fritz and Kowalski, have enabled the first fully automatic detection, identification, spectroscopic classification, and public reporting of a transient: SN 2023tyk. This milestone represents a boost in efficiency for BTS and an image of what time-domain astronomy could look like during the Rubin era. It also hints toward the discovery potential of adapting similar technology to other areas of time-domain astronomy.

Acknowledgments

We thank the anonymous reviewer for the thoughtful and constructive comments, which contributed to this manuscript.

A great number of people have contributed to BTS and BTS scanning over the years. We thank the following people who have saved 10 or more sources to internal BTS catalogs on Fritz as of 2023 October: Ivan Altunin, Raphael Baer-Way, Pallas A. Beddow, Ofek Bengiat, Joshua S. Bloom, Aleksandra Bochenek, Emma Born, Kate Bostow, Victoria Mei Brendel, Rachel Bruch, Vidhi Chander, Matthew Chu, Elma Chuang, Aishwarya Dahiwale, Asia deGraw, Dmitry Duev, Kingsley Ehrich, Eli Gendreau-Distler, Nachiket Girish, Xander Hall, K-Ryan Hinds, Ido Irani, Cooper Jacobus, Connor Jennings, Joel Johansson, Snehaa Ganesh Kumar, Michael May, William Meynardie, Shaunak Modak, Kishore Patra, Neil Pichay, Sophia Risin, Yashvi Sharma, Gabrielle Stewart, Nora Linn Strotjohann, James Sunseri, Edgar Vidal, Jacob Wise, Abel Yagubyan, Yoomee Zeng, and Erez A. Zimmerman.

We also thank Jakob Nordin for discussions relating to AMPEL.

Some spectra used for classifying sources in the training set were collected with the MMT Observatory. MMT Observatory and Zwicky Transient Facility access was supported by Northwestern University and the Center for Interdisciplinary Exploration and Research in Astrophysics (CIERA).

The material contained in this document is based upon work supported by a National Aeronautics and Space Administration

(NASA) grant or cooperative agreement. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect the views of NASA. This work was supported through a NASA grant awarded to the Illinois/NASA Space Grant Consortium.

This research was supported in part through the computational resources and staff contributions provided for the Quest high performance computing facility at Northwestern University, which is jointly supported by the Office of the Provost, the Office for Research, and Northwestern University Information Technology.

Based on observations obtained with the Samuel Oschin Telescope 48 inch and the 60 inch Telescope at the Palomar Observatory as part of the Zwicky Transient Facility project. ZTF is supported by the National Science Foundation under grants No. AST-1440341 and AST-2034437 and a collaboration including current partners Caltech, IPAC, the Oskar Klein Center at Stockholm University, the University of Maryland, University of California, Berkeley, the University of Wisconsin at Milwaukee, University of Warwick, Ruhr University, Cornell University, Northwestern University, and Drexel University. Operations are conducted by COO, IPAC, and UW.

SED Machine is based upon work supported by the National Science Foundation under grant No. 1106171.

The Gordon and Betty Moore Foundation, through both the Data-Driven Investigator Program and a dedicated grant, provided critical funding for SkyPortal.

N.R. and A.A.M. are partially supported by LBNL subcontract No. 7707915.

M.W.C. acknowledges support from the National Science Foundation with grant Nos. PHY-2308862 and PHY-2117997.

S.R.K. thanks the Heising-Simons Foundation for supporting his research.

Facilities: PO:1.2 m, PO:1.5 m

Software: Astropy (Astropy Collaboration et al. 2013, 2018, 2022), corner (Foreman-Mackey 2016), Jupyter (Kluyver et al. 2016), Keras (https://keras.io), Matplotlib (Hunter 2007), NumPy (Harris et al. 2020), pandas (McKinney 2010; The pandas development team 2024), penquins (Duev et al. 2021), scikit-learn (Pedregosa et al. 2011), SciPy (Virtanen et al. 2020), SkyPortal (van der Walt et al. 2019; Coughlin et al. 2023), Tensorflow (Abadi et al. 2016), tqdm (da Costa-Luis 2019), and the Weights and Biases platform (https://wandb.ai/site/).

Appendix A
Accuracy and Loss Evolution during Training

Figure 11 shows how the accuracy and loss evolve during the training of the final BTSbot models, including the production model. Differences in how the $N_{\max}$ alert thinning is applied between train and validation (see Section 2) and the use of class weights during training can explain much of the difference between their respective loss curves. Thus, the train loss being significantly lower than validation loss cannot be interpreted as overfitting. There does appear to be overfitting in the production model (bold curves) past epoch ∼30, evidenced by the validation loss beginning to increase while the train loss continues to decrease. We avoid this by selecting the model that produced the minimum validation loss during training, rather than the model from the final epoch of training. Figure 11 marks the epochs where the learning rate decreased due to a plateau in the validation loss as vertical gray lines. The learning rate decreases three times, spanning from $\alpha = 10^{-4}$ to $\alpha = 6 \times 10^{-6}$. We also observe that the selected production model is not the model that produced the least validation loss or had the greatest validation accuracy. The production model is selected by considering the best performance in policy-based metrics, those most relevant to how BTSbot is used.

Training this production model took ∼32 hr on an Intel Xeon 6230 CPU, and an inference on a single alert packet takes less than a hundredth of a second on a laptop CPU.

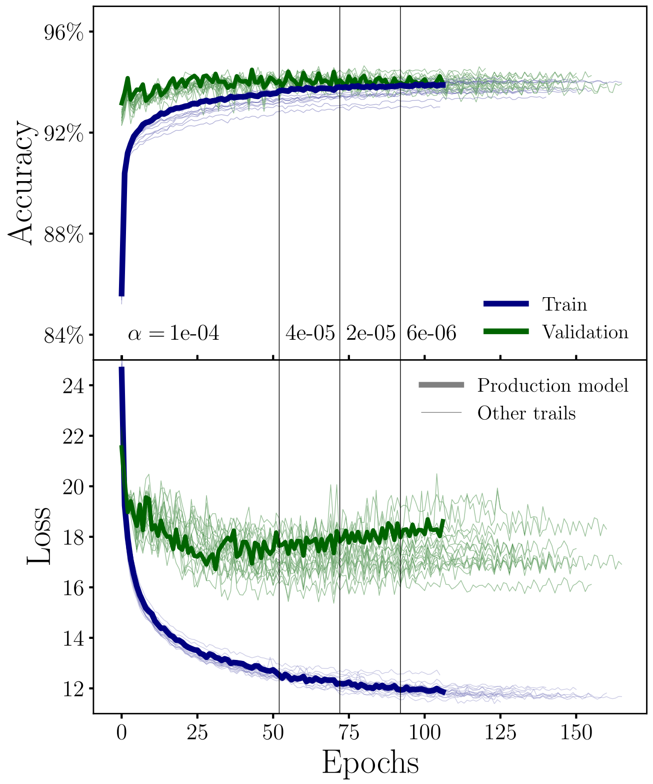


Figure 11. Accuracy (top) and loss (bottom) for the final BTSbot models (production model, bold curve; other trials, narrow curves) of the train (blue) and validation (green) splits during training as a function of epoch. Gray vertical lines indicate epochs at which the learning rate decreased. The discrepancies in these metrics between the train and validation splits cannot be interpreted as overfitting alone because they are partially due to differences in alert thinning and class weighting.

Appendix B
Integration of BTSbot into ZTF and the BTS Workflow

BTSbot has been deployed into ZTF's first-party alert broker, Kowalski, to enable running in real-time on incoming alert packets from IPAC's alert-producing and brokering system. Kowalski performs three distinct operations on every incoming alert packet. First, it separates the alert packet from its `prv_candidates` field (a 30 day history of detections and nondetections) to concatenate it to the full list of `prv_candidates` from all previous alerts with the same `objectId`. This forms a full light curve for a given ZTF object. The product of this concatenation is then used to compute the custom metadata features BTSbot takes as input (see Section 2 for a list and respective definitions). Kowalski then crossmatches every alert with a large number of catalogs such as NED,[31] the Census of the Local Universe

[31] https://ned.ipac.caltech.edu/

(Cook et al. 2019), Milliquas (Flesch 2019), and others. Information from these crossmatches is made available to scanners and automated processes. Lastly, Kowalski runs several ML models: braai, the five ACAI classifiers (Duev & van der Walt 2021), and BTSbot. The outputs of all models are injected into the alert packet along with their corresponding version numbers allowing alert filters to use this information when identifying candidate transients. The enriched alert packets are then stored in a nonrelational database (MongoDB[32]), allowing users to design custom, potentially complex, filters as pure database queries for maximum flexibility. These filters, including the BTS alert filter, run on every alert.

If an alert passes a filter, it will be sent as a candidate to Fritz, a SkyPortal instance that serves as the ZTF collaboration's Marshal. Two additional layers of filtering are applied to allow the automated saving and triggering of any instrument for which SkyPortal has a corresponding API. The first filtering layer assesses if the candidate should be saved, and the second assesses whether an instrument should be triggered for follow-up. Autotriggering is only run on sources that have passed the autosaving filter. In the case of BTSbot, two autosaving and autotriggering filters are implemented, one for each of the policies bts_p1 and bts_p2.

Additional features are implemented in SkyPortal to minimize redundant triggers and erroneous saves. Some science teams direct their scanners to maintain lists of sources that frequently pass their alert filter but are not of interest, e.g., the *junk* set maintained by BTS scanners (see Section 2). These can optionally be used to prevent any automatic action on the source. A new, automatic follow-up request is prevented if there is a source within 0.5" that already has (i) a spectrum, (ii) a classification, (iii) a pending follow-up request for the same instrument, or (iv) been cataloged by BTS in *junk*. These criteria have been tuned over months of sending BTSbot triggers in real-time to minimize the number of FP triggers. Instruments like SEDM, which can conduct both spectroscopic and photometric observations, get additional rules to define whether the new trigger is redundant with an existing trigger. The payload used for triggering an instrument, which contains the triggering instructions like the priority for SEDM, is set in advance alongside the Kowalski autotriggering filter, but the priority assigned to a target can be dynamically increased as new alerts are posted. In particular, a source triggered on after passing bts_p1 can have its payload updated to priority 2 if it passes bts_p2 before the trigger is completed.

When autosaving or autotriggering actions are taken, comments recording these actions are posted to the relevant source page on Fritz. Beyond bookkeeping, this is crucial to facilitate scanners working alongside BTSbot by displaying BTSbot's actions in the same interface where manual scanning is performed. Seamless integration with the tools that scanners already rely on enables joint working that is more efficient and more reliable than either separately. This also allows scanners to modify the triggers sent by BTSbot, for example, increasing the trigger's priority or adding photometry to the request. These features make real, automated triggers safer and more dependable, thus making the automated aspects of BTS require less monitoring. The results from the SEDM observations and associated data products are uploaded back to Fritz for visualization and, typically, classification.

[32] https://www.mongodb.com

Appendix C Comparison with Unimodal Architectures

An exploration of simpler, alternative architectures and a characterization of their performance is necessary to justify BTSbot's chosen multimodal architecture. We present alert- and policy-based performance metrics produced by two unimodal model architectures: a unimodal convolutional neural network (UM-CNN) and a fully connected NN. These networks perform an inference using only images and only metadata respectively. The training data, image preprocessing, and feature extraction are, where relevant, performed identically for all three architectures. The ordering and types of layers of the unimodal architectures are identical to those of the multimodal architecture with the other branch entirely removed. The hyperparameters of the layers, the Adam optimizer, and the value of $N_{\max}$ are searched over in a Bayesian hyperparameter sweep similarly to the sweeps executed for the MM-CNN.

Table 5 shows the optimal layer hyperparameters for either architecture. Compared to the MM-CNN's convolutional branch, the UM-CNN has smaller convolutional kernels and a much larger dense layer following the flattening. The UM-CNN's first two convolutional layers have more filters than the final two convolutional layers, whereas the pattern is reversed for the MM-CNN. We also find that the optimized UM-CNN has a batchsize = 32, and Adam exponential decay rate parameters $\beta_1 = \beta_2 = 0.999$. Compared to the MM-CNN's metadata branch, the optimized NN has half as many neurons in the first two dense layers but 8 times as many in the third layer. We also find that the NN performs best with a batchsize = 128, Adam parameters $\beta_1 = \beta_2 = 0.999$, and $N_{\max} = 30$. The hyperparameter values that are not explicitly

Table 5
Architectures of Unimodal BTSbot Alternatives

Unimodal Convolutional Neural Network		Fully Connected Neural Network	
Layer Type	Layer Parameters	Layer Type	Layer Parameters
2D convolution	64 filters, 3 × 3 kernel	Batch normalization	⋯
2D convolution	64 filters, 3 × 3 kernel	Dense	64 units
Max pooling	2 × 2 kernel	Dropout	0.40
Dropout	0.45	Dense	64 units
2D convolution	16 filters, 3 × 3 kernel	Dense	64 units
2D convolution	16 filters, 3 × 3 kernel	Dropout	0.70
Max pooling	4 × 4 kernel	Dense	1 unit
Dropout	0.65	...	
Flattening	⋯	...	
Dense	128 units	...	
Dropout	0.45	...	
Dense	1 unit	...	

Note. Layer parameters of alternative BTSbot architectures: unimodal convolutional neural network (left two columns) and fully connected neural network (right two columns). Optimal hyperparameters are determined by large hyperparameter sweeps.

Table 6
Performance Metrics of Unimodal BTSbot Alternatives

Architecture Type	Alert-based Metrics		Policy-based Metrics				
	Accuracy (%)	ROCAUC	Completeness (%)	bts_p1 Purity (%)	bts_p2 Purity (%)	Δt_{save} (days)	$\Delta t_{\mathrm{trigger}}$ (days)
NN	95.5	0.988	99.8	82.8	91.2	−0.0400	−0.0200
UM-CNN	82.4	0.902	94.3	69.5	92.5	−0.0485	−0.0174
MM-CNN	94.9	0.985	100.0	84.6	93.0	−0.0381	−0.0147

Note. Comparison of performance metrics on test split data across three different model architectures: a fully connected neural network (NN), a unimodal convolutional neural network (UM-CNN), and a multimodal convolutional neural network (MM-CNN). Each network is the highest performing in at least one metric. The UM-CNN is less complete than the MM-CNN, and both the NN and the UM-CNN fall short of the MM-CNN in purity. Overall, the MM-CNN is the best performing architecture.

mentioned can be assumed to be identical to that of the MM-CNN.

Table 6 compares test split performance metrics from each of the three architectures. The NN slightly outperforms the MM-CNN in the alert-based metrics while the UM-CNN falls well short of the others. The UM-CNN presented here uses the full-size image cutouts, but Appendix D shows that slightly increased performance can be realized through the reduction of the input image's size. In policy-based metrics, the MM-CNN's marginal advantage in completeness over the NN is due to having just a single fewer FN. The MM-CNN demonstrates a larger advantage in purity, however. Its advantage is ∼0.5%–2% over the others in bts_p2 purity; even small boosts in purity are valuable when applied over the large samples and baselines in consideration by BTS. Each model performs very similarly in the speed metrics, although the NN and the UM-CNN marginally outperform the others in $\Delta t_{\mathrm{trigger}}$ and Δt_{save} respectively.

Each architecture outperforms the others in some metrics, but the MM-CNN delivers the best overall performance. It delivers the highest completeness and purity while sacrificing well less than an hour in $\Delta t_{\mathrm{save/trigger}}$.

Despite having worse purity than the MM-CNN, there is still valuable utility in the NN because we find that it trains ∼60 times faster than the MM-CNN. This discrepancy in training time is partially due to the NN adopting $N_{\max} = 30$ instead of $N_{\max} = 100$ as the MM-CNN does, and could be also decreased by training the MM-CNN on a GPU. These factors aside, the NN would very likely remain many factors quicker to train, and it is thus better suited for experimenting with adapting BTSbot to alternative use cases. In developing adaptations of BTSbot, being able to very quickly design and execute experiments will be key. Excluding images may also lend advantages in the ease of transferring BTSbot adaptations to other surveys, for example, LSST or the upcoming La Silla Schmidt Southern Survey (LS4[33]). Although effort has been made to develop survey-agnostic CNNs (e.g., Cabrera-Vives et al. 2023), BTSbot is not designed or expected to perform consistently on image data from other surveys. Instead, one could more easily design an NN that exclusively uses survey-agnostic features, increasing its potential impact by allowing it to be applied more widely.

[33] https://sites.northwestern.edu/ls4/

Appendix D Convolutional Neural Networks with Cropped Image Cutouts

The properties of the image cutouts produced by large wide-field surveys are critical to a number of the survey's functions. While the pixel scale, typically measured in arcseconds (″) per pixel, is a fixed property of the telescope's optics, many choices can be made in software that determine the nature of the cutouts sent to alert brokers. One must choose the angular and pixel size of the cutouts in light of the pixel scale and whether or not to decrease the resolution by binning the pixels. A tension arises because scanners generally prefer having more information, i.e., cutouts with higher resolution and greater angular size, but the surveys producing and disseminating the alerts prefer minimal network bandwidth costs, i.e., lower resolution and smaller angular size. Smaller cutouts with less information may compromise the performance of these key ML models, however, possibly reducing the scientific potential of the entire survey. Only a small number of studies developing CNNs for large wide-field surveys comment on performance over a range of cutout sizes (e.g., Carrasco-Davis et al. 2021; Killestein et al. 2021; Reyes-Jainaga et al. 2023). It remains to be seen, then, what the minimum cutout size is to maintain acceptable performance with current CNNs. Adding further complexity, CNNs performing different tasks will be impacted differently by reduced cutout sizes, and MM-CNNs may be more resilient to reduced cutout sizes due to the availability of metadata information. Multiscale cutouts, where resolution progressively decreases moving away from the cutout center, are a compelling alternative because they dramatically increase the available field of view without an increase to the data volume of the cutouts (Reyes-Jainaga et al. 2023). Further study of (MM-)CNN performance over a range of cutout sizes is necessary to better characterize their correlation.

To explore this important issue, we train multiple UM-CNN versions of BTSbot to identify which cutout size produces the most accurate model. We create new training sets from BTSbot's original training set by keeping only the innermost $N_{\mathrm{pix}} \times N_{\mathrm{pix}}$ pixels of each cutout and renormalizing them individually. We train ∼30–60 trials of BTSbot with a given value of N_{pix} (3, 5, 7, 13, 21, 35, or 49) in Bayesian hyperparameter sweeps similar to Section 3.1 and Appendix C. These trials are unlikely sufficient to yield optimal hyperparameters, but they do give a representative view of the performance for each value of N_{pix}. For models with

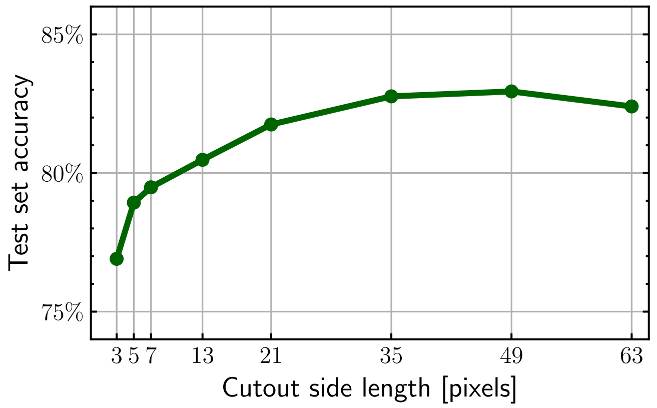


Figure 12. Test set accuracy of BTSbot UM-CNNs as a function of input image cutout size. Very small cutouts (3 × 3 to 13 × 13 pixels) clearly underperform relative to models with larger cutouts (35 × 35 to 63 × 63 pixels). The highest performing model is notably not that which uses the full-size images but rather the model that uses 49 × 49 pixel cutouts. Cropping cutouts could allow for the significant decrease of a survey's data rate and possibly network bandwidth costs, in this case, with no decrease in the performance of ML-based transient detection tools.

$N_{\rm pix} \leqslant 13$, we remove the pooling layers entirely to maintain the image's size through the hidden layers.

Figure 12 presents the highest performing models judged by accuracy on the test set. We find that the $N_{\rm pix} = 49$ UM-CNN yields the best performance, and that both the $N_{\rm pix} = 49$ and $N_{\rm pix} = 35$ UM-CNNs marginally outperform the UM-CNN with uncropped cutouts ($N_{\rm pix} = 63$). It is not unambiguously clear what the reason for this is from this simple experiment alone; however, these results suggest that information $\sim 0\rlap{.}'5$ away from the source in question tends to be irrelevant or noisy when performing our task. We notably do not observe the same trend for the MM-CNN, where the $N_{\rm pix} = 63$ model is the highest performing. Carrasco-Davis et al. (2021) find that the ALeRCE stamp classifier (an MM-CNN), performs best with $N_{\rm pix} = 21$ cutouts. Together, this indicates that more work is necessary to better understand the optimal cutout size for upcoming surveys.

The $N_{\rm pix} = 35$ model roughly matches the $N_{\rm pix} = 63$ model in accuracy but has $35^2/63^2 \approx 30\%$ the number of pixels. In this case, shrinking cutouts dramatically quickens BTSbot's training without compromising performance. Shrinking cutouts survey-wide is an option to reduce a survey's network bandwidth costs significantly as image cutouts typically comprise a large fraction of the bytes in an alert packet. Although 35 × 35 cutouts are appropriate for BTSbot, such small cutouts may reduce scanners' ability to distinguish sources of different types. Experiments of this sort are most relevant to the alert stream design of upcoming surveys like LSST and LS4. LSST's planned alert packet cutout size is at least 30 × 30 pixels (Graham et al. 2024) at a pixel scale of $0\rlap{.}''2$ pixel^{-1} (Ivezić et al. 2019), thus spanning 6″ on a side. The $N_{\rm pix} = 5$ and $N_{\rm pix} = 7$ models are nearest to this in terms of the field of view; however, ZTF's much larger pixel scale (1″ pixel^{-1}) makes this an unreasonable comparison. Further study is required to assess how small cutouts can be without compromising, e.g., real/bogus performance at LSST-like resolutions. LS4 will have a pixel scale very similar to that of ZTF, but, at the time of writing, the alert packet cutout size is undetermined (R. Knop 2024, private communications). These results provide the preliminary guidance that LS4 cutouts should be no smaller than 21 × 21 in order to maintain the performance of BTSbot-like UM-CNN models.

ORCID iDs

Nabeel Rehemtulla https://orcid.org/0000-0002-5683-2389
Adam A. Miller https://orcid.org/0000-0001-9515-478X
Theophile Jegou Du Laz https://orcid.org/0009-0003-6181-4526
Michael W. Coughlin https://orcid.org/0000-0002-8262-2924
Christoffer Fremling https://orcid.org/0000-0002-4223-103X
Daniel A. Perley https://orcid.org/0000-0001-8472-1996
Yu-Jing Qin https://orcid.org/0000-0003-3658-6026
Jesper Sollerman https://orcid.org/0000-0003-1546-6615
Ashish A. Mahabal https://orcid.org/0000-0003-2242-0244
Russ R. Laher https://orcid.org/0000-0003-2451-5482
Reed Riddle https://orcid.org/0000-0002-0387-370X
Ben Rusholme https://orcid.org/0000-0001-7648-4142
Shrinivas R. Kulkarni https://orcid.org/0000-0001-5390-8563

References

Abadi, M., Agarwal, A., Barham, P., et al. 2016, arXiv:1603.04467
Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, ApJ, 935, 167
Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Bailey, S., Aragon, C., Romano, R., et al. 2007, ApJ, 665, 1246
Bellm, E. C., Kulkarni, S. R., Barlow, T., et al. 2019a, PASP, 131, 068003
Bellm, E. C., Kulkarni, S. R., Graham, M. J., et al. 2019b, PASP, 131, 018002
Blagorodnova, N., Neill, J. D., Walters, R., et al. 2018, PASP, 130, 035003
Bloom, J. S., Richards, J. W., Nugent, P. E., et al. 2012, PASP, 124, 1175
Boone, K. 2019, AJ, 158, 257
Bostroem, K. A., Pearson, J., Shrestha, M., et al. 2023, ApJL, 956, L5
Breiman, L. 2001, Mach. Learn., 45, 5
Brink, H., Richards, J. W., Poznanski, D., et al. 2013, MNRAS, 435, 1047
Cabrera-Vives, G., Bolivar, C., Förster, F., et al. 2023, Machine Learning for Astrophysics, 40th Int. Conf. on Machine Learning, 5
Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., & Maureira, J.-C. 2017, ApJ, 836, 97
Carrasco-Davis, R., Reyes, E., Valenzuela, C., et al. 2021, AJ, 162, 231
Carrasco Kind, M., & Brunner, R. J. 2013, MNRAS, 432, 1483
Cold, C., & Hjorth, J. 2023, A&A, 670, A48
Cook, D. O., Kasliwal, M. M., Sistine, A. V., et al. 2019, ApJ, 880, 7
Coughlin, M. W., Bloom, J. S., Nir, G., et al. 2023, ApJS, 267, 17
da Costa-Luis, C. 2019, JOSS, 4, 1277
de Soto, K. M., Villar, A., Berger, E., et al. 2024, arXiv:2403.07975
Dekany, R., Smith, R. M., Riddle, R., et al. 2020, PASP, 132, 038001
Dieleman, S., Willett, K. W., & Dambre, J. 2015, MNRAS, 450, 1441
Domínguez Sánchez, H., Huertas-Company, M., Bernardi, M., Tuccillo, D., & Fischer, J. L. 2018, MNRAS, 476, 3661
Duev, D., Shin, K. M., & Singer, L. 2021, dmitryduev/penquins: a python client for dmitryduev/kowalski, v2.1.2, Zenodo, doi:10.5281/zenodo.5651471
Duev, D. A., Mahabal, A., Masci, F. J., et al. 2019, MNRAS, 489, 3582
Duev, D. A., & van der Walt, S. J. 2021, arXiv:2111.12142
Flesch, E. W. 2019, OJAp, 6, 49
Foreman-Mackey, D. 2016, JOSS, 1, 24
Förster, F., Cabrera-Vives, G., Castillo-Navarrete, E., et al. 2021, AJ, 161, 242
Fremling, C., Hall, X. J., Coughlin, M. W., et al. 2021, ApJL, 917, L2
Fremling, C., Miller, A. A., Sharma, Y., et al. 2020, ApJ, 895, 32
Fukushima, K., & Miyake, S. 1982, PatRe, 15, 455
Gagliano, A., Contardo, G., Foreman Mackey, D., Malz, A. I., & Aleo, P. D. 2023, ApJ, 954, 20
Goldstein, D. A., D'Andrea, C. B., Fischer, J. A., et al. 2015, AJ, 150, 82
Gomez, S., Berger, E., Blanchard, P. K., et al. 2020, ApJ, 904, 74
Gomez, S., Villar, V. A., Berger, E., et al. 2023, ApJ, 949, 113
Goobar, A., Johansson, J., Schulze, S., et al. 2023, NatAs, 7, 1137
Graham, M. J., Kulkarni, S. R., Bellm, E. C., et al. 2019, PASP, 131, 078001

Graham, M. L., Bellm, E., Guy, L., et al. 2024, LSST Alerts: Key Numbers DMTN-102, Vera C. Rubin Observatory, https://dmtn-102.lsst.io/
Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Natur, 585, 357
Hendrycks, D., & Gimpel, K. 2016, arXiv:1610.02136
Hiramatsu, D., Tsuna, D., Berger, E., et al. 2023, ApJL, 955, L8
Hosseinzadeh, G., Dauphin, F., Villar, V. A., et al. 2020, ApJ, 905, 93
Hunter, J. D. 2007, CSE, 9, 90
Irani, I., Prentice, S. J., Schulze, S., et al. 2022, ApJ, 927, 10
Itagaki, K. 2023, TNSTR, 2023-1158
Jacobson-Galán, W. V., Dessart, L., Margutti, R., et al. 2023, ApJL, 954, L42
Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, ApJ, 873, 111
Kaiser, N., Aussel, H., Burke, B. E., et al. 2002, Proc. SPIE, 4836, 154
Kasliwal, M. M., Cannella, C., Bagdasaryan, A., et al. 2019, PASP, 131, 038003
Killestein, T. L., Lyman, J., Steeghs, D., et al. 2021, MNRAS, 503, 4838
Kim, Y. L., Rigault, M., Neill, J. D., et al. 2022, PASP, 134, 024505
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, in Positioning and Power in Academic Publishing: Players, Agents and Agendas, ed. F. Loizides & B. Schmidt (Amsterdam: IOS Press), 87
Lanusse, F., Ma, Q., Li, N., et al. 2018, MNRAS, 473, 3895
LeCun, Y., Bengio, Y., & Hinton, G. 2015, Natur, 521, 436
Mahabal, A., Rebbapragada, U., Walters, R., et al. 2019, PASP, 131, 038002
Masci, F. J., Laher, R. R., Rusholme, B., et al. 2019, PASP, 131, 018003
McCulloch, W. S., & Pitts, W. 1943, Bull. Math. Biophys., 5, 115
McKinney, W. 2010, Proc. 9th Python in Science Conf., ed. S. van der Walt & J. Millman, 56
Miranda, N., Freytag, J. C., Nordin, J., et al. 2022, A&A, 665, A99
Möller, A., & de Boissière, T. 2020, MNRAS, 491, 4277
Morgan, R., Nord, B., Bechtol, K., et al. 2022, ApJ, 927, 109
Morgan, R., Nord, B., Bechtol, K., et al. 2023, ApJ, 943, 19
Muthukrishna, D., Narayan, G., Mandel, K. S., Biswas, R., & Hložek, R. 2019, PASP, 131, 118002
Nair, V., & Hinton, G. E. 2010, Proc. 27th Int. Conf. on Machine Learning (Madison, WI: Omnipress), 807, https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf
Nordin, J., Brinnel, V., van Santen, J., et al. 2019, A&A, 631, A147
The pandas development team 2024, pandas-dev/pandas: Pandas, Zenodo, doi:10.5281/zenodo.3509134
Pasquet, J., Bertin, E., Treyer, M., Arnouts, S., & Fouchez, D. 2019, A&A, 621, A26
Patterson, M. T., Bellm, E. C., Rusholme, B., et al. 2019, PASP, 131, 018001
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, JMLR, 12, 2825
Perley, D. A., Fremling, C., Sollerman, J., et al. 2020, ApJ, 904, 35
Perley, D. A., Gal-Yam, A., Irani, I., & Zimmerman, E. 2023, TNSAN, 119, 1
Qin, Y.-J., Zhang, K., Bloom, J., et al. 2023, arXiv:2309.10022
Qu, H., Sako, M., Möller, A., & Doux, C. 2021, AJ, 162, 67
Rehemtulla, N. 2024, BTSbot v10 training set, v10, Zenodo, doi:10.5281/zenodo.10839691
Rehemtulla, N., Miller, A., du Laz, T., et al. 2024, nabeelre/BTSbot: Publication Version, v1.0.1, Zenodo, doi:10.5281/zenodo.10839685
Rehemtulla, N., Miller, A., Fremling, C., et al. 2023a, TNSAN, 265, 1
Rehemtulla, N., Miller, A. A., Coughlin, M. W., & Jegou du Laz, T. 2023b, arXiv:2307.07618
Reyes-Jainaga, I., Förster, F., Muñoz Arancibia, A. M., et al. 2023, ApJL, 952, L43
Rigault, M., Neill, J. D., Blagorodnova, N., et al. 2019, A&A, 627, A115
Rodríguez, Ó., Maoz, D., & Nakar, E. 2023, ApJ, 955, 71
Sadeh, I., Abdalla, F. B., & Lahav, O. 2016, PASP, 128, 104502
Shappee, B. J., Prieto, J. L., Grupe, D., et al. 2014, ApJ, 788, 48
Sharma, Y., Sollerman, J., Fremling, C., et al. 2023, ApJ, 948, 52
Sharon, A., & Kushnir, D. 2022, MNRAS, 509, 5275
Simonyan, K., & Zisserman, A. 2014, arXiv:1409.1556
Smith, K. W., Smartt, S. J., Young, D. R., et al. 2020, PASP, 132, 085002
Sollerman, J., Yang, S., Perley, D., et al. 2022, A&A, 657, A64
Stein, R., Mahabal, A., Reusch, S., et al. 2024, ApJL, 965, 10
Stoppa, F., Bhattacharyya, S., Ruiz de Austri, R., et al. 2023, A&A, 680, 16
Szegedy, C., Zaremba, W., Sutskever, I., et al. 2013, arXiv:1312.6199
Tachibana, Y., & Miller, A. A. 2018, PASP, 130, 128001
Tonry, J. L. 2011, PASP, 123, 58
Tonry, J. L., Denneau, L., Heinze, A. N., et al. 2018, PASP, 130, 064505
Tsaprazi, E., Jasche, J., Goobar, A., et al. 2022, MNRAS, 510, 366
Turpin, D., Ganet, M., Antier, S., et al. 2020, MNRAS, 497, 2641
van der Walt, S. J., Crellin-Quick, A., & Bloom, J. S. 2019, JOSS, 4, 1247
van Roestel, J., Duev, D. A., Mahabal, A. A., et al. 2021, AJ, 161, 267
Villar, V. A., Berger, E., Miller, G., et al. 2019, ApJ, 884, 83
Villar, V. A., Hosseinzadeh, G., Berger, E., et al. 2020, ApJ, 905, 94
Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, NatMe, 17, 261
Walmsley, M., Ferguson, A. M. N., Mann, R. G., & Lintott, C. J. 2019, MNRAS, 483, 2968
Walmsley, M., Smith, L., Lintott, C., et al. 2020, MNRAS, 491, 1554
Wright, D. E., Smartt, S. J., Smith, K. W., et al. 2015, MNRAS, 449, 451
Yang, S., Sollerman, J., Strotjohann, N. L., et al. 2021, A&A, 655, A90
Zimmerman, E. A., Irani, I., Chen, P., et al. 2024, Natur, 627, 759