

ATCAT (@🐱): Astronomical Timeseries CAusal Transformer

zora tung,¹

¹*Independent researcher.*

PREPRINT ONLY, not yet submitted, PLEASE CHECK BACK FOR UPDATES

ABSTRACT

The Legacy Survey of Space and Time (LSST) at the Vera C. Rubin Observatory will capture light curves (LCs) for 10 billion sources and produce millions of transient candidates per night, necessitating scalable, accurate, and efficient classification. To prepare the community for this scale of data, the Extended LSST Astronomical Time-Series Classification Challenge (ELAsTiCC) sought to simulate a diversity of LSST-like time-domain events. Using a small transformer-based model and refined light curve encoding logic, we present a new state of the art classification performance on ELAsTiCC, with $71.79 \pm 0.28\%$ F1 on LC-only classifications, and $89.75 \pm 0.04\%$ F1 on LC+metadata classifications. Previous state of the art was $65.5 \pm 0.28\%$ F1 for LC-only, and for LC+metadata, 84% F1 with a different setup and $83.5 \pm 0.6\%$ F1 with a directly comparable setup. Our model outperforms previous state-of-the-art models for fine-grained early detection at all time cutoffs, which should help prioritize candidate transients for follow-up observations. We demonstrate label-efficient training by removing labels from 90% of the training data (chosen uniformly at random), and compensate by leveraging regularization, bootstrap ensembling, and unsupervised pretraining. Even with only 10% of the labeled data, we achieve $67.44 \pm 0.06\%$ F1 on LC-only and $87.07 \pm 0.18\%$ F1 on LC+metadata, validating an approach that should help mitigate synthetic and observational data drift, and improve classification on tasks with less labeled data. We find that our base model is poorly calibrated via reliability diagrams, and correct it at a minimal cost to overall performance, enabling selections by classification precision. Finally, our GPU-optimized implementation is $9\times$ faster than other state-of-the-art ELAsTiCC models, and can run inference at ~ 33000 LCs/s on a consumer-grade RTX 4090 GPU, making it suitable for large-scale applications. It is also correspondingly cheaper to train, making it accessible to more researchers.

Key words: methods: data analysis – methods: statistical – software: machine learning – software: public release – supernovae: general – stars: variables: general

1 INTRODUCTION

The LSST at the Vera C. Rubin Observatory offers the potential to uncover millions of transient events, both to improve our understanding of known objects and to find novel ones (LSST Science Collaborations 2009). However, it is challenging to identify these transients, because most new ones are faint, single-pixel sources. Additionally, the scale of data requires the use of automated techniques (Fraga et al. 2024). Classification systems are used to distinguish interesting transient objects from variable stars and less interesting ones, to allow for immediate follow-up spectroscopy, further optical photometry, and imaging in other wavebands. They are also used for population studies, such as improving the mapping of the Milky Way by identifying more RR Lyrae, and detecting a large number of supernovae to study the dependence of dark energy properties on direction. Astronomers also hope to find rare objects and events such as novae and stellar flares, gamma-ray bursts and X-ray flashes, active galactic nuclei, stellar disruptions by black holes, and evidence of neutron stars and black hole binaries; some of these require timely follow-up, and others exist for a longer time, but are difficult to identify (see Željko Ivezić et al. 2008 for an overview). The quality of classification systems determines how much value can be obtained through timely follow-up observations, and the quality of downstream scientific analyses. This motivates our work.

Classification models for time-domain astronomy encounter several challenges, which set them apart from traditional machine

learning (ML) models such as image classification or natural language processing. In the ELAsTiCC dataset, which is synthetic but carefully designed to mimic real data, light curves not only have extreme values and occasionally feature high observational noise, but are also irregularly sampled, and each “channel” (band-pass color filter) is sampled at a different time. For example, a given object may have a flux value in the u band one day, and then an observation in the g band 2 days later; this corresponds to physical color filter plates being changed on the telescope. Objects may only be visible during part of the year. In this manner, the problem may appear more in the domain of traditional statistics, but well-tuned ML models tend to perform better, likely reflecting the high-dimensional, complex nature of the underlying objects being modeled.

Cabrera-Vives et al. (2024) demonstrated the viability and superiority of neural network approaches for ELAsTiCC, by both creating a strong random forest (RF) baseline model, and outperforming it with a 3-layer transformer model (ATAT). Several notable classifiers exist in the space of LSST classification; in 2018, the (Photometric LSST Astronomical Time-Series Classification Challenge) (PLAsTiCC) Kaggle challenge was run; resulting classification models include Boone (2019) and Qu et al. (2021). The ELAsTiCC dataset is largely seen as its successor, and classification models for it include Fraga et al. (2024); Cabrera-Vives et al. (2024); Shah et al. (2025); Gupta et al. (2025a); Moreno-Cartagena et al. (2025). Model accuracy was

a key focus of the Kaggle competition and remains a primary focus. However, it is not enough to have a high F1 score; in order to be useful, a model needs to adapt to real LSST data, be fast enough for large datasets, achieve strong performance for early detection, provide useful embeddings on out-of-domain classes, and give good uncertainty estimates.

Machine learning use for transient classification has a long history (Bailey et al. 2007; Bloom et al. 2012; Long et al. 2012; Lochner et al. 2016; Charnock & Moss 2017; Naul et al. 2018; Carrasco-Davis et al. 2019; Gómez et al. 2020; Jamal & Bloom 2020), including recent efforts to classify SN types and variables (Muthukrishna et al. 2019; Villar et al. 2019; Möller & de Boissière 2019; Boone 2021; Qu et al. 2021; Gagliano et al. 2023; Pimentel et al. 2023; Donoso-Oliva et al. 2023; Rehemtulla et al. 2024; Rizhko & Bloom 2024). Very recent efforts include Li et al. (2025) who explore using models such as Moirai (Woo et al. 2024) and Chronos (Ansari et al. 2024), finding good performance from time series models, despite those models being trained on data from much different domains. Tan et al. (2025) and Donoso-Oliva et al. (2025) apply transformer-style models to the MACHO LC classification task. Vidal et al. (2025) applies a transformer-based model to synthetic MOSFiT light curves, estimating simulator parameters.

Machine learning systems often incur problems with domain adaptation (please see Koh et al. 2020 for an excellent overview) and poor uncertainty estimates, which can mean that their accuracy comes at the expense of these secondary objectives. These problems are well-known, but there is no scientific or industrial consensus on how to address them; practitioners must examine a variety of methods, each with their own drawbacks. Significant discrepancies between ELAsTiCC and observational data are expected. The effective domain for the problem may gradually shift over time as well, as telescope conditions and scientific understandings of the distribution/classification of various objects change, and may benefit from techniques such as active learning (Richards et al. 2011) or online learning (see Mohri & Medina 2012 for ML theory). Machine learning systems also exhibit issues with generalization (a subtly different problem than domain adaptation, defined as lower accuracy with a new sample from the same domain, commonly seen as divergence in training/validation performance), and calibration. But at least with ELAsTiCC, these are more easily addressed with standard techniques, as we demonstrate later.

We develop and evaluate several techniques to improve accuracy, early detection, calibration, and label efficiency. We developed a light curve encoding technique that significantly increases classifier accuracy over previous state of the art. We leverage our decoder architecture to train towards good early detection performance. We effectively calibrate our model, and discuss the benefits of calibration while cautioning that it does not resolve model bias. We do this while effectively leveraging consumer-grade hardware for fast and cost-effective training and inference. We also achieve strong performance when removing labels from 90% of the ELAsTiCC dataset, demonstrating label-efficient training. This should make our model useful for application to other, smaller datasets, as well as facilitate use of techniques such as active learning for ameliorating domain adaptation issues as these models are applied to LSST observational data.

We experiment with pretraining via a generative objective, predicting the next flux value at a given time and wavelength (and flux_err). This is similar to the objective used by NLP models such as language models. The usefulness, bias, quality, and ethical implications of LLMs are contested (Bender et al. 2021; Lee et al. 2023; Perrigo 2023; Raji et al. 2022), but they nevertheless exhibit a surprising level of generalization given their pretrain-

ing objective (predicting the next word from previous ones), and this pretraining is absolutely essential to model quality. Moirai (Woo et al. 2024) has demonstrated that unsupervised learning can be effectively leveraged for time-domain data, to produce better forecasting estimates than several alternatives. In astronomical contexts, Zhang et al. (2024); Rizhko & Bloom (2024) both use the CLIP constrastive objective (Radford et al. 2021) to align spectra and light curves, with Zhang focusing on transients and Rizhko and Bloom focusing on variables. These objectives are unsupervised and quite valuable, but geared towards high-accuracy simulation data, whereas our pretraining attempts to learn from the light curves directly, and we intend to pretrain on [unlabeled] LSST observational data once that is available. Gupta et al. (2025a,b) explore transfer learning from a simulated ZTF dataset to the simulated ELAsTiCC dataset. While this exercises reuse of deep neural network embeddings, it does not use an unsupervised objective.

The LSST will produce vast quantities of unlabeled data (and previous efforts such as the ZTF already have), and making use of it is desirable. In addition to some accuracy boost, unsupervised pretraining can improve generalization and domain adaptation (Erhan et al. 2010; Goyal et al. 2021). While all observational data is biased by which objects are bright enough to be detectable, unsupervised data should have less selection bias than labeled datasets. Unsupervised pretraining should also make embeddings more generally useful, since the objective incentivizes the last layer embedding to maintain information that may not matter for classification. Our setup also allows for generative modeling. While in astronomical contexts we have physical simulators which generate quite meaningful light curves, our model may be useful for reproducing observational phenomena that are not yet well captured by the simulations, and providing an alternative preprocessing method for interpolation to a new time / channel grid.

In this work, we introduce ATCAT (@🐱), a classification model that considerably improves on previous state of the art. In many ways, it builds on the excellent work of ATAT (Cabrera-Vives et al. 2024). Our code and models are currently released at <https://atcat.click>. In Section 2, we describe our methods. In Section 3, we present the results of our model, including ablations that help explain which of its improvements were most significant. In Section 4, we re-articulate the scientific value of our model, and discuss many future directions which could be taken with this work. For readers interested in the encoding method which provides the largest ELAsTiCC F1 scores improvement, key results are presented in Section 2.2.1 and 3.5.1.

2 METHODS

In this section, we present our approaches for modeling photometric light curves, primarily for the end goal of classification. A photometric light curve consists of a sequence of observations, each consisting of a time, channel (band pass color filter), flux error (provided by instruments and/or upstream models), and a “calibrated flux” value. This flux value has mean field subtraction applied, which for our purposes means that it can be negative.

For classification purposes, our model takes this light curve as input and produces a weighted distribution of class labels. For generative and data augmentation purposes, we generally think of time, channel (band pass color filter), and flux error values as being arbitrary and given to us, but the flux reflecting the true nature of the object. We consider its conditional distribution $P_{t,c}$

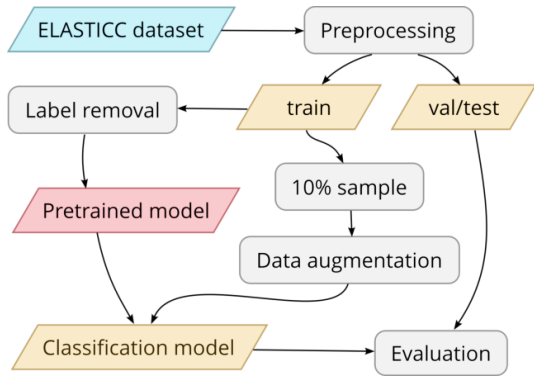


Figure 1. High-level schematic of dataflow for pretraining and fine-tuning (training of the classification model). In implementation, all grey nodes except preprocessing are run at runtime, with instrumentation to double-check their correctness. See text.

as a Gaussian, with standard deviation given by flux error,

$$\begin{aligned} \text{flux_err}_{t,c} &\sim P_{\text{err}} \text{ (arbitrary distribution across the dataset)} \\ \text{flux}_{t,c} &\sim \mathcal{N}(\text{true_flux}_{t,c}, \text{flux_err}_{t,c}^2) =: P_{t,c} \end{aligned} \quad (1)$$

For all models (except ablation studies), we replace the integer channel index (representing 6 color filters from ultraviolet to infrared, called u, g, r, i, z, Y) with the central wavelength of each color filter / band pass. We will refer to this as “channel wavelength”.

An overview system diagram for the unsupervised pretraining and fine-tuning setup is in Figure 1. The full-label training dataflow is similar: simply replace the “10% sample” node with a pass-through and remove the “label removal” and “pretrained model” nodes.

2.1 The ELASTICC dataset

We train and evaluate our model on the ELASTICC dataset, a realistic simulation of LSST time-series data. Example light curves are shown in Figure 3 (more examples in the Appendix).

2.1.1 ATAT splits and labels

We use the ELASTICC v1 dataset (Malanchev 2023). Our preprocessing is similar to ATAT (Cabrera-Vives et al. 2024) and involves:

- The same 20-way classification scheme as ATAT, by combining some of the original 32 classes. Please refer to (Cabrera-Vives et al. 2024) Figure 2 for a class distribution frequency and motivation.
- For each object (grouped by SNID), we find t_{alert} , when the alert flag is present, and include observations (light curve points) up to 30 days before this alert flag. Times are reset so that $t = 0$ is the time of the first point.
- The exact same training / validation / test splits as ATAT, as they have shared the exact IDs used in each split. The data is first split into a test distribution with 20,000 examples (1,000 for each class), and the remainder as 5-way cross validation (so each split has 80% train and 20% val).
- For metadata, we train a QuantileTransformer with `output_distribution="normal"` using scikit-learn, similar to ATAT. We save nan/null and $\pm\infty$ bits, and add separate embedding vectors when those bits are active.

Our preprocessing is very similar to ATAT to facilitate 1-1 comparisons, although we use Polars for better CPU efficiency. The only difference in semantics is that for metadata, ATAT replaces

nan/null/ $\pm\infty$ values with -9999, whereas we preserve the semantic distinction between these values, which may improve robustness when data has both missing and extreme values.

2.1.2 Data augmentation

To reduce overfitting to the training data, we implemented several data augmentation processors. These routines were stacked upon one another, so it was possible for multiple (or all) augmentation functions to be applied to a single light curve. The routines include,

- **Subsampling:** selects a random subset of points, retaining at least 10 points. The exact number of points retained was chosen uniformly between 10 and the number of points minus 1. It was randomly applied at a rate of 25%, although we passed through LCs with 10 or fewer points unmodified, so its effective rate is a bit lower.
- **Flux scaling:** scales `flux` and `flux_err` by a uniform random value in $[\frac{1}{1.1}, 1.1] \approx [0.909, 1.1]$. Each point in a light curve is scaled by the same value. It was randomly applied at a rate of 20%.
- **Time scaling:** shifts the time of all points by a uniform random value in $[0, 10]$. Time shifting should help ensure the model is robust if an alert is flagged a bit earlier or later. It was randomly applied at a rate of 20%.
- **Redshifting:** modifies channel wavelengths. We sampled $z_{\text{additional}} \in \mathcal{N}(0, \sigma^2)$ with $\sigma = 0.1$, and then clipped this to a min value of -0.1 (slightly blue-shifted). Then we multiplied each channel wavelength by $(1 + z_{\text{additional}})$. This augmentation was randomly applied at a rate of 20%.
- **Random noise:** samples an additional flux error term $e_{i,j}$ for LC point j of example i , and then samples the actual error $\xi_{i,j} \sim \mathcal{N}(0, e_{i,j}^2)$. $e_{i,j}$ is added to `flux_err`, and $\xi_{i,j}$ is added to `flux`. $e_{i,j}$ ’s come from a scaled uniform distribution $\text{Unif}[0.002\sigma, 0.2\sigma]$, where $\sigma = \text{std}(\text{flux}_i)$. If we only have one flux value, we replace the standard deviation term by 1. *This augmentation is not applied to our main models*, but for all of our pretraining setups except the baseline, it was randomly applied at a rate of 15%.

Ablation studies can be found in Section 3.5.2.

2.2 Model architecture

Our model consists of encoders for metadata and light curves, a transformer with 4 layers, and an output classifier, as shown in Figure 2. Our base architecture is a transformer, similar to Vaswani et al. (2017). As ATAT (Cabrera-Vives et al. 2024) has demonstrated, transformers are quite capable models, outperforming models based on feature extraction and random forests (RF). Many other architectures have been attempted (see Introduction), including convolutional networks and RNNs. The transformer is popular in part because of the attention mechanism’s ability to effectively share information at various time scales, and in part because it is relatively GPU-friendly.

We follow ATAT’s settings for the main model dimension, and for the rest of this section we denote

d = model dimension (default 384)

We used 4 attention heads, and a relatively small 64-dim attention (so 16-dim key / query / value vectors per attention head). We omit bias terms in key / query / value attention projections. Our feed-forward units in each transformer layer had twice the embedding dimension (768). Model sizes are the result of light tuning.

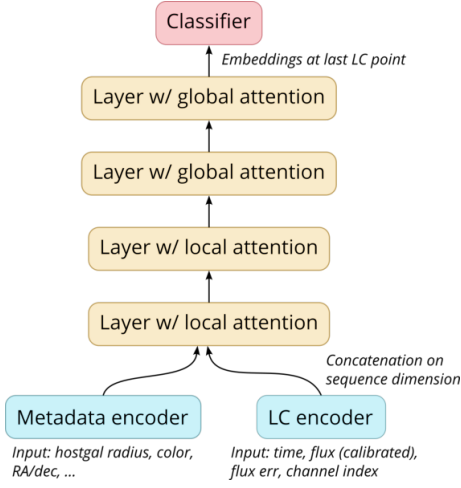


Figure 2. Our model architecture. Our model is a transformer with 4 layers. The first two layers use local attention, the first with a 1-day threshold, and the second with a 10-day threshold. Metadata is encoded as the first token in the sequence, allowing LC points in the global attention layers to attend to it.

2.2.1 Light curve encoding

One major challenge of dealing with irregularly-sampled data like ELAsTiCC is encoding the values in an efficient manner. Convolutions at various time-scales are a popular choice. Some of these like the fourier transform involve basis functions over all time values, and others such as wavelets or convolutional neural network kernels, use windowed basis functions.

Following ATAT, we attempt to embed each LC point as a single sequence element in the transformer. Let i index into examples in our dataset and j index into the LC points of the i th example (there are differing numbers of points per example), then

$$x_{i,j} \in \mathbb{R}^4 \sim \text{time, flux, flux_err, channel wavelength}$$

Each of these 4-dimensional vectors will be embedded into a d -dimensional vector.

Information is preserved if we can embed unique $x_{i,j}$ samples to unique embeddings. Our approach has advantages compared to convolutional or interpolation approaches, which might either choose a grid that is too fine or too coarse. Grids that are too coarse can lose information. Grids that are too fine add extra sequence elements, which in our model would end up causing an $O(n^2)$ cost in the attention mechanism. Furthermore, it is common for examples to be lacking measurements in one channel wavelength (this occurs for ~18% of ELAsTiCC examples), and much less likely but still possible that they only have measurements in a single channel wavelength. Our approach appears better here, since it does not introduce noisy estimations for these missing values.

A key concern is whether the network is actually able to extract usable information from the embeddings, i.e. learn to compute reasonable functions from it. We found that that the large set of dynamic (LC-based) features extracted by ATAT no longer improved ATCAT performance, validating the legitimacy of our approach, and suggesting that complex feature extraction is not necessary for ELAsTiCC classification.

Our input encoding routine consists of the following steps,

(i) We embed each time value $t \in \mathbb{R}$ into a d -dimensional vector

$$\text{concat}([\sin(\alpha_k t)]_k, [\cos(\alpha_k t)]_k)$$

Letting $m = d/2$ and $k \in 0..m-1$, we define $\alpha_k = 10^{-T_{\max} \cdot \beta_k}$, where β_k 's are essentially the concatenation of two $m/2$ -

dimensional vectors $\text{linspace}(-0.1, 0)$ and $\text{linspace}(0, 1)$, except without repeating the point at 0. In this manner, our α_k term is very similar to the term from Vaswani et al. (2017), but instead of taking β_k 's as $[0, 1/m, \dots, m-1/m]$, we choose time-scales from -0.1 , capturing higher-frequency signals. Indeed, Δt between a pair of points in ELAsTiCC can be much smaller than 1, and so it makes sense to have some frequencies (in our case, half of them) $\alpha_k > 1$ where these sin and cos values will be separated. By contrast, Vaswani et al. (2017) embeds positional indices in place of our t values, which are integers (their “ Δt ” is 1). We use $T_{\max} = 1500$ following ATAT.

(ii) We scale the time values by 1/10 to keep their L_2 norm around 1, and then pass them through a linear transformation.

(iii) We embed flux and flux_err with a float-value embedder. Each value $x \in \mathbb{R}$ is mapped to a small 4-dimensional vector $[\tanh(s_k x)]_k$ where $s_k \in [1, 10, 1000, 100000]$, and then this is mapped through a `nn.Linear` layer to the embedding dimension (384 by default). The `nn.Linear` (affine) layer does have a bias term, but it is zero-initialized. We chose these functions so that the network could effectively have terms that vary linearly within a certain dynamic range. Our multi-scale tanh approach allows us to preserve dynamic range while avoiding outlier saturation of activations, which residual networks have difficulty recovering from.

(iv) We rotate all of the time embeddings using the rotary position encoder Su et al. (2021). We choose scales α_i as

$$\log \alpha_i = 4 \sin\left(\frac{2\pi i}{d} - 0.5\right)^{10} - 4 \quad i \in 0..\text{model_dim} - 1$$

We chose this analytic function visually, such that we could experiment with rotations by both wavelength and time (shifting the 0.5 offset), but our final models do not use this. We do not believe it is better than the default rotary position encoder. We normalize the input “index” (here, channel wavelength) to effectively rotate each pair of values by $\alpha_i \frac{\lambda}{2}$ where $\frac{\lambda}{2} \in [0, 1]$ except when red-shifting beyond LSST wavelengths. Empirically, we found that rotating only flux or flux_err embeddings was worse, and rotating all channels is about the same. We hypothesize that key-query comparisons in the attention are gathering information on different time-scales, and these rotations allow them to gather such information with a more or less strong preference for matching the channel wavelength.

Ablation studies can be found in Section 3.5.1, future work in 4.6.

2.2.2 Metadata encoding

We encode metadata by feeding the quantile scores for 85 features into a basic embedder, consisting of a base $d \rightarrow d$ affine transformation, and a residual nonlinearity consisting of a $d \rightarrow d$ affine transformation, leaky ReLU, and another $d \rightarrow d$ affine transformation. Both affine transformations connected to the input are given a slight L1 penalty, to encourage the model to ignore features which do not provide a strong signal.

2.2.3 Packing input sequences

For training classification models, we packed the metadata as a single input token at the beginning, and then added embeddings for light curve points. The index of the last light curve point is used to index into the last layer for classification, using `torch.gather`, because each element in a batch of light curves may have a different sequence length.

Encoding values for our pretrained model was more interesting. After each light curve point, we included the time, channel wavelength, and flux error for the next point. One batch element might look like:

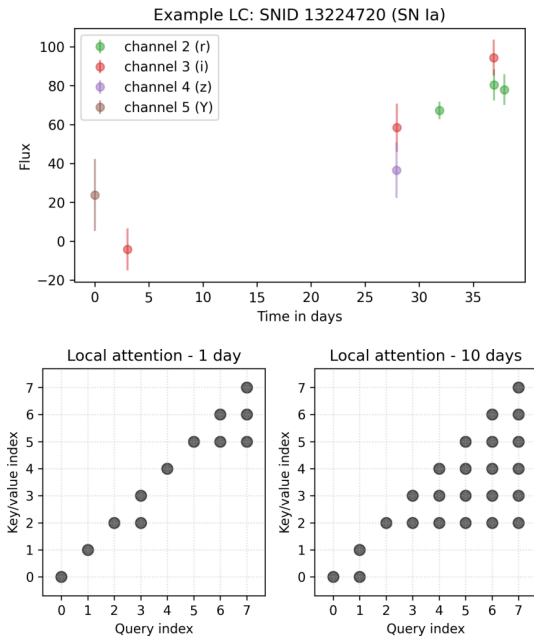


Figure 3. Local attention connections. We visualize the connections in our local attention mechanism for a specific ELAsTiCC example (this one is a Type-Ia supernova). The first transformer layer, featuring a local attention mechanism with a threshold of 1 day, allows the two points in channels 3 (i) and 4 (z) near $t = 27$ to attend to each other, and likewise the cluster of 3 points at the end (near $t = 37$), but only causally (bottom left figure). For the first pair, this is represented by the point at query index 3 being allowed to attend to key/value index 2 (presence of a black dot). For the second layer, with a threshold of 10 days (bottom right figure), the first two points can attend to each other, and all of the other points as well, but also only causally.

[META] [LC0] [PRED1] [LC1] [PRED2] ...

Then, for training the generative model, we again used a gather operation to retrieve all of the [PREDICT] sequence elements. In terms of implementation, generating these indices can be tricky; we found it effective to use sequential CPU code (per batch element), since the total amount of computation is small compared to the model.

2.2.4 Transformer with local attention

Our main transformer stack, shown in Figure 2, is a 4-layer hybrid transformer. The first two layers have a local masking scheme in their attention mechanism. This means that each LC point is only allowed to attend to points which are before it and within a time range. For example, if our threshold is 10 (days), then a point with $t = 31$ can attend to a point with $t = 25$ but not a point with $t = 20$. An example of this mechanism is in Figure 3. For local transformer layers, sequence elements containing metadata could not attend to sequence elements for any light curve points. Our model is fully causal, also known as a decoder architecture. Causal masking means that any light curve point can only attend to those before it. We do not believe causal masking provided any performance boost over full attention, but did make it easier to train early detection and generative models.

2.2.5 Classification

We use a very simple classification layer, consisting of a layer norm, and affine projection to the number of output classes. For most of our models, we also added an extra output class, which we intended to correspond to “unsure”. Scores were calculated as a softmax (or often log-softmax) including this class. However,

we found that even with Soft F1 losses (see next section), the models generally would only learn to put very minimal weight on this class, learning to output a smoothed distribution over other classes. We did use this extra output class for calibration (Section 2.7) and novel object detection (Section 4.3).

2.3 Evaluation and losses

For most of this work, we are reporting macro F1 scores. Please see [Opitz & Burst \(2019\)](#) for a precise technical definition. In general, for all classes, we will compute

$$P_i = \frac{\#TP \text{ (true pos)}}{\#TP + \#FP \text{ (false pos)}} \quad R_i = \frac{\#TP}{\#TP + \#FN \text{ (false neg)}}$$

and then, the harmonic mean of P_i, R_i and then take a simple average over classes. For our evaluation metrics, we use the implementation in `torchval.metrics`.

2.3.1 Class-balanced metrics for the val dataset

We implemented a version of the F1 metric that is less sensitive to class imbalances early on, and argue that it contributed greatly to the integrity of our experiments. The val and test splits differ greatly in their distributions of exemplars, and therefore using a metric which in expectation is the same for both means that we didn’t need to look at any test metrics until this writeup.

Macro F1 alone does not fix class imbalance. Suppose we have n examples of P_i and n examples of another class $P_{i'}$, and the classifier confuses these classes at some rate. If we add 10x more elements of our “other” class $P_{i'}$, even if they are from the same distribution, the FP count of P_i will be greatly affected. On ELAsTiCC with ATAT’s splits, this effect is very strong. We provide a very concise explanation of our class-balanced F1 score, loosely following the notation of [Opitz & Burst \(2019\)](#). Formally, let X be the space of input features, $Y = 1..n$ be our label set, $f : X \rightarrow Y$ be our classifier, S be our dataset consisting of (x, y) pairs, and $\{m_{ij}\}_{i \in 1..n, j \in 1..n}$ be our weighted confusion matrix,

$$m_{ij} = \sum_{x, y \in S} w_{x, y} \mathbf{1}_{f(x)=i \text{ and } y=j}$$

where $w_{x, y} \propto \frac{1}{|\{x', y' \in S : y'=y\}|}$ weights by inverse class frequency, and $\mathbf{1}_{\text{cond}}$ is an indicator function. Then define

$$P_i = \frac{m_{ii}}{\sum_{y=1}^n m_{iy}} \quad R_i = \frac{m_{ii}}{\sum_{y=1}^{n+1} m_{yi}}$$

and proceed as normal.

2.3.2 Soft/trainable F1 scores

We also implemented a version of soft F1 scores for training. These can be defined by considering our model’s outputs as a vector of scores (formally, $f : X \rightarrow [0, 1]^Y$), and writing

$$m_{ij} = \sum_{x, y \in S} w_{x, y} f(x)_i \mathbf{1}_{\{y = j\}}$$

(In our training setups, we don’t need to worry about the weight term, because we use a balanced batch sampler.)

2.4 Training routine

We used Pytorch default initializers (Kaiming for linear/affine transformations), except for float value embedders as mentioned previously. Our models which had full access to training data did not have dropout, but for ensembled models for smaller labeled sets (our unsupervised pretraining experiments), we used 15% dropout for transformer layers, and 4% for input and classifiers.

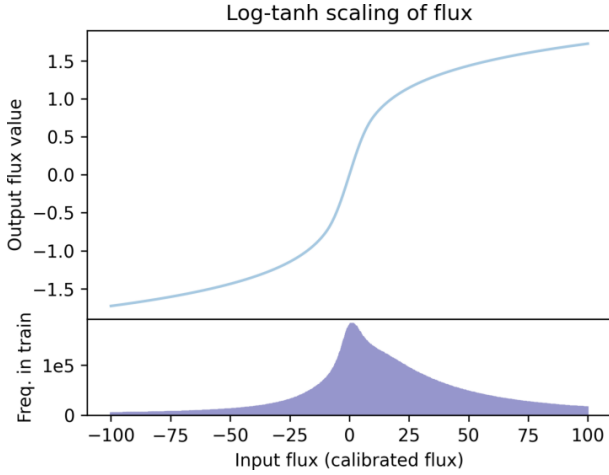


Figure 4. Nonlinear scaling of flux values for generative modeling. We squash the flux values to a much smaller range, by “gluing” together a tanh function (around 0) and log function, matching the first derivative and intercept point. We also pre-scale flux by 1/10. This keeps the response curve not too flat for the majority of values, while scaling the max value from 2,568,897 to 6. The first part of the figure is the response curve, the bottom is a histogram (aligned in X-axis values) of all training flux values from all light curves (not showing the long tail of extreme values). As elsewhere, we are looking at calibrated flux values after mean field subtraction, which can be negative.

We did not see much difference whether dropout was placed before or after each attention/feed-forward unit, and have them after by default.

We used a training regime similar to ATAT, using an nadam optimizer with $2e-4$ learning rate, $\beta_1 = 0$ and $\beta_2 = 0.999$, which is similar to RMSProp. We used a learning rate schedule with a linear warm-up for the first 1000 steps, steady $2e-4$ learning rate, then exponential decay after 20,000 steps with a half-life of 6,000 steps. For pretraining, we randomly shuffle training data; for fine-tuning / labeled training, we used ATAT’s balanced batch sampling approach, which yields batches that are balanced among labels. For setups with less data, we enabled the nadam weight decay term with weight $1e-5$ as well. During pretraining, we clip gradients to a L2 norm of 0.1, but we do not have any gradient clipping for fine-tuning / labeled training.

For the setup with the full training dataset, we trained for 40,000 steps, evaluating every 4,000 steps. For the main results, we used our class-balanced F1 metrics on the validation set to select the ideal checkpoint, but for several sub-experiments such as early detection, calibration, and generation of confusion matrices, we picked step 36,000 for implementation convenience. Generally, the model performance would change by at most 0.14% after step 32,000. For our unsupervised pretraining setups, we pretrained models for 20,000 steps, and then fine-tuned for 20,000 steps.

2.5 Unsupervised pretraining

As we briefly explained in the introduction, our goal with unsupervised pretraining is to capture information about the distributions of light curves in the model parameters, so that it will generalize better when trained on a limited amount of supervised data. In order to mimic this setup for ELASrTiCC, we removed labels from 90% of the training data. We found it much more convenient and equally performant to run unsupervised pretraining on all training data, and then fine-tune on the 10% of training data with labels, although we also tried mixing these objectives.

We experimented with two objectives:

- Predicting values: This is the dominant approach for LLMs.

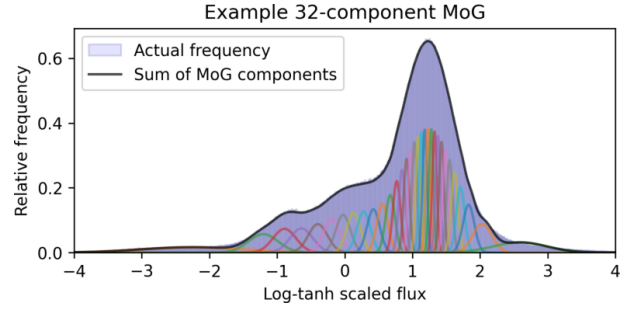


Figure 5. Mixture-of-Gaussian components for generative modeling. We graph the components of our mixture of Gaussians model, showing actual frequencies of scaled flux values, and the sum of Gaussian components (black line, each component equally weighted). For our actual models, we used 64 and 128-component models; here we show only 32 for ease of visualization. The black line matches our histogram fairly well, as desired, with some deviation around ~ 2.4 for this 32-component model. Incorrect choices of sigmas (component width) will result in the black line being jagged, or components being too broad. Individual components are the many colored lines.

Unlike LLMs, we chose to model the conditional distribution $p(\text{flux}|\text{time}, \text{flux_err}, \text{channel wavelength})$, but not model the distribution of the subsequent time, channel wavelength, or flux_err values.

- Patch prediction from full transformer attention: this approach was used by Moirai (Woo et al. 2024). We masked out contiguous ranges of points between 5% and 50% of the total range of time values, with hand-tuned logic to select ranges of data that appeared informative (details omitted, since we did not proceed with this approach).

Preliminary experiments showed that the two approaches had similar value for the purpose of fine-tuning classification models. The first approach is simpler and more data efficient, so we ended up using it. Please see Section 2.2.3 for how we formatted these inputs to our model; for the [PREDICT] elements, we used the same embedding technique as Section 2.2.1, omitting the flux component and adding a free parameter (so that the model could learn to distinguish token types).

2.5.1 Nonlinearity

For unsupervised pretraining, we deal with extreme values in the ELASrTiCC dataset slightly differently than in feature encoding, since our approach in feature encoding (of combining multiple scalings) cannot be used here. We ended up squashing the target flux values with a nonlinearity f . This function is shown in Figure 4, and is similar to the nflows codebase (Durkan et al. 2020). We projected flux to $f(\text{flux})$, and

$$\text{scaled flux_err} = \frac{f(\text{flux} + \text{flux_err}) - f(\text{flux} - \text{flux_err})}{2} \quad (2)$$

Recall that we are thinking of modeling the distribution of true flux with observational error, $P_{t,c}$ as in (1). We don’t have access to true $\text{flux}_{t,c}$ and want to work in the projected space (with less extreme values), so we instead conceptualize

$$P'_{t,c} = \mathcal{N}\left(f(\text{scaled flux}_{t,c}), \sigma^2 = f(\text{scaled flux_err}_{t,c})^2\right)$$

This makes little sense statistically, and we think that experimenting with Moirai’s approach of using heavier-tailed distributions (e.g. t-distributions) would be valuable future work (see Woo et al. 2024). For the goal of getting useful hidden embeddings for downstream use, how principled we are may not be of utmost importance. For generative modeling, it is more important; our

model won't fail to run, but it won't capture several phenomena that it should.

We expressed our predicted next flux distribution $Q_{t,c}$ as mixture weights for a Mixture-of-Gaussians model, where the means and sigmas are fixed. These fixed values come from the training data with a very simple algorithm; we chose our n means and sigmas as

$$\begin{aligned} \vec{\text{means}}(n) &= \text{quantile}(\vec{\text{flux}}, \text{linspace}(1/2n, 1 - 1/2n, n)) \\ \vec{\text{sigmas}}(n) &= 0.7 \cdot \text{diff}(\vec{\text{means}}(n + 1)) \end{aligned}$$

where the monospace-font functions are from torch or numpy. Visually, this seemed to match our distribution well; see Figure 5. We also tried tuning the means and sigmas by maximizing the log-likelihood of samples of flux values, but found that this only increased noise.

In order to generate a predicted next flux distribution in the original space, we can project the means and standard deviations back, since our nonlinearity f is invertible. We use the same approach for flux_err as (2), simply replacing f with f^{-1} . Our approach fails to represent the most extreme points. However, it solved a practical problem we encountered trying to predict unscaled flux values: the model would often learn to use the very wide mixture components at the extremes rather than ones with much closer means, because the Gaussian tails fall off too quickly. We hypothesize that Moirai's approach would fix this issue.

2.5.2 Predicting flux with observational noise

Since the values we are trying to predict in ELAsTiCC include observational error, we have a few natural choices for how to evaluate our model. Do we care about forward KL divergence,

$$\text{KL}(P'_{t,c} || Q_{t,c})$$

or reverse KL, $\text{KL}(Q_{t,c} || P'_{t,c})$? Our insight should derive from the fact that for distributions A and B , if B is broader than A , then

$$\text{KL}(A || B) < \text{KL}(B || A)$$

For reverse KL, when the model is quite confident (say, it just got a low-noise LC point in the same band), then it won't get harshly penalized by a point with high observational noise. However, when observational noise is low, but there just isn't enough information, then the model will get a huge penalty. In practice, these huge penalties end up overtaking the loss, and we have to artificially soften $P'_{t,c}$ to learn anything at all.

But forward KL is not perfect either, especially in the case we just mentioned when the model is confident and correct, but we have a lot of observational noise. The same applies to logloss / δ_{flux} . Our solution is to instead think about modeling

$$Q_{t,c,\text{err}} \sim \text{flux}_t | t, c, \text{flux_err}_{t,c}, (\text{previous points})$$

and the model can therefore learn to increase its variance if we have a high $\text{flux_err}_{t,c}$.

There is a third possibility, log-loss " $q_{t,c}(\text{flux}_{t,c})$ ", denoting the density $q_{t,c}$ [which can also be conditioned on error, for $q_{t,c,\text{err}}$]. By taking limits or relaxing formalities appropriately, this is equal to $\text{KL}(\delta_{\text{flux at } t,c} || Q_{t,c})$ where $\delta_{\text{flux at } t,c}$ is the Dirac delta, similar to forward KL. It appears to not assume $P'_{t,c}$ is normal, which could be a considerable advantage, at the cost of potentially being less efficient, because it will only learn the variance of $P'_{t,c}$ from seeing many samples over our dataset, instead of directly from $\text{flux_err}_{t,c}$. We leave experimentation as future work.

2.5.3 Generative performance evaluation

In Section 3.2, we compare generative performance with the Gaussian Process used by Boone (2019), projecting our distributions of predicted flux back to the original unscaled space, as described in Section 2.5.1. We studied forward KL divergence on 1000 test set examples, by first filtering out sequences with less than 6 points, and then randomly selecting 50 from each class. This was a smaller study partly because the GP was expensive to run in an iterative-decoding fashion, since it needs to be re-trained for each new LC point. For both models, we start measuring prediction performance on the 5th point; this resulted in predictions on 61,099 LC points in total. Since the Gaussian process was only predicting a single Gaussian, we used the simpler closed form for its KL divergence. For our model, we had to sample the KL divergence, and drew 4000 samples for each point. To ensure that we had drawn enough samples to accurately compute the KL, we subsampled this sample (using 5-way CV-like folds), and ensured that the standard deviation of subsample KL scores was small (its median is 0.005).

2.6 Early detection

In order to improve early detection performance, we simply added a logloss which is evaluated at many LC points instead of just the final one (we skipped the first point on the assumption it may be too noisy). We averaged this loss across sequence length per example first, so that long examples would not get disproportionately more loss than short ones. We noticed that final prediction accuracy decreased with only this loss, and so we compensated by adding back our old last-point loss (in the form of logloss, not Soft F1) with 1/3 of the early detection loss weight. This approach makes good use of our causal decoder architecture, effectively training towards early detection performance at all time values simultaneously.

2.7 Calibrated models

We evaluated several methods to calibrate our model, including

- (i) Simple label smoothing, following Section 7 of Szegedy et al. (2015)
- (ii) Focal loss (Lin et al. 2017), including a variant where we removed the loss re-weighting from the gradient calculation
- (iii) Training the unsure logit towards inverse train-time accuracy on each example
- (iv) Training a bootstrap ensemble, and then training the unsure logit towards the bootstrap ensemble's accuracy on each example

Label smoothing adds a ϵ/n to the true probabilities for each example (subtracting from the label class' probability); we choose $\epsilon = 0.1$ following Szegedy et al. (2015).

Focal loss (Lin et al. 2017) re-weights examples by their difficulty. For a single example, suppose q_{true} is the output probability of the true class.

$$w(q_{\text{true}}) = (1 - q_{\text{true}})^\gamma \quad \text{FL}(q_{\text{true}}) = -w(q_{\text{true}}) \log(q_{\text{true}})$$

We tried $\gamma \in \{0.5, 1, 2\}$. We tried detaching calculations in w from automatic gradient calculation, and found that this consistently provided a slight improvement. For our setup, $\gamma = 0.5$ seemed sufficient to get good calibration results, and had the best F-1 scores.

We also tried a runtime technique where we try to have the "unsure" logit (see Section 2.2.4) probabilities to correspond to accuracy of each batch at training time (to compute accuracy, we compute the mean over batch elements where the logit with

the highest output probability is correct, detaching tensors from automatic gradient calculation). Let b index over batch elements, and $q_{b,\text{true}}$ be the output probability of the true class for batch element b . We first compute a smoothed “confident and correct” score, and from this create an “unsure” score for each batch element,

$$\text{cc}_b = \text{avg} (q_{b,\text{true}}, \min(0, 2(q_{b,\text{true}} - 1/2)))$$

$$\text{unsure}'_b = \frac{0.1}{N_{\text{classes}}} + (1 - \text{cc}_b)$$

$$\text{unsure}_b = \frac{1 - \text{accuracy}}{\text{mean}_b(\text{unsure}')} \text{unsure}'_b$$

and then smooth the actual label probabilities, so the true label gets $1 - \text{unsure}_b$ weight and the unsure label gets unsure_b weight; we train towards this with KL divergence. Effectively, we train the model so that it will output “unsure” at a rate corresponding to its accuracy, but allowing it to be quite confident on some examples and less confident on others.

Finally, we tried a bootstrapping method, using a first round of classifiers to estimate how difficult each example is. We first train 4 bootstrap sub-models, each of which gets 1/2 of the data. We minimize the maximum overlap between these models, a fun mini mathematical puzzle, where one divides the training into 6 partitions and assigns 3 of these to each model. We intentionally under-train this for 12k steps and regularize at fairly high 20% dropout, $1e-4$ weight decay, and then ensemble these models together (adding probability scores) and run it on the training to generate per-class probabilities for each class. We do keep all sub-models in the ensemble, so half of them will have seen a particular example as training input; given the early stopping and high regularization, we presume that it hasn’t overfit terribly. We take the probability of the true class as our p_{true} and one minus this as our p_{unsure} , and add a loss to train the classifier’s logits towards this using KL divergence. We keep the base soft F1 loss towards just the true label to lose too much overall quality.

3 RESULTS

3.1 Main results

In Table 3.1 we show the main results for state of the art ELAsTiCC classification models. The third model, a vision transformer from [Moreno-Cartagena et al. \(2025\)](#), is referred to as the name of its base model “SWINv2” in text, so we repeat that here. These results generally show our model as a considerable improvement over ATAT and the SWINv2 model. ORACLE was evaluated on ELAsTiCC v2 with a smaller number of classes, but the differences between ELAsTiCC v1 and v2 are unclear, due to lack of published articles on the matter. The ORACLE team makes comparisons against ATAT, suggesting they believe ELAsTiCC v1 and v2 numbers are comparable, but a more 1-1 comparison would be strongly preferred. Unfortunately, ELAsTiCC preprocessing is tricky. While ORACLE has also shared their preprocessing routines, the lack of standardization means we could not immediately integrate it. We have made some suggestions for improving this situation, which would facilitate more accurate model comparisons and lead to better scientific understandings (see Section 4.8). For model sizes: ATAT’s model sizes were not reported, but we loaded a checkpoint and computed the total number of parameters for all tensors. The SWINv2 seems to be based on the “Tiny” variant; the v2 paper is not open access, but the v1 paper lists this variant as having 29M parameters [Liu et al. \(2021\)](#).

In Table 2, we show fine-class performance of ATCAT. Please see Figure 9 for confusion matrices. [Shah et al. \(2025\)](#) notes that ATAT ([Cabrera-Vives et al. 2024](#)) underperforms on CARTs

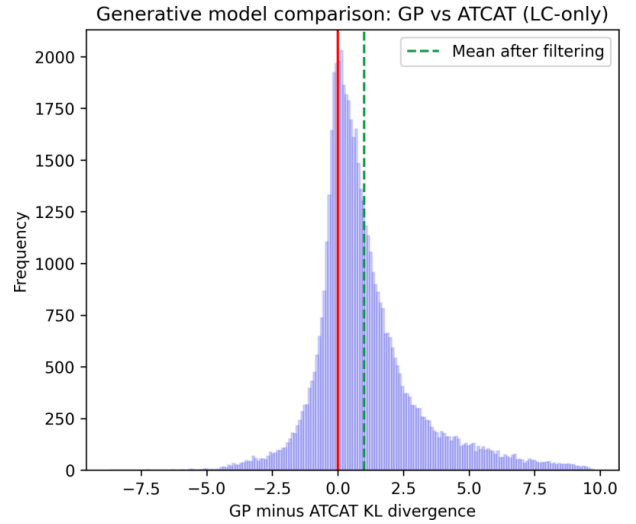


Figure 6. Evaluation of generative modeling performance. We predict next LC points for a sample of 1000 light curves (50 per class) on the test dataset. For each point, we compute the difference of forward KL divergence; assuming that the model’s flux is $P = \mathcal{N}(\text{flux}, \text{flux_err})$ for the point in question, we then look at the predicted distribution of flux Q by both of these models, and then compute $\text{KL}(P||Q_{\text{GP}}) - \text{KL}(P||Q_{\text{ATCAT}})$. Remember that KL divergence is always non-negative and smaller values are better; when the KL is zero then the model matches the observation perfectly. Hence, when this difference is positive (right of the solid red line), our model is better, and when it is negative, the GP is better. See text for details.

compared to the random forest, but our model has reversed that weakness and outperforms both the RF (random forest) and ATAT on CART, although it is still one of the more difficult classes. Please see [Shah et al. \(2025\)](#) for a valuable discussion on CART classification. Our model achieves leading F1 scores on all fine classes, but sometimes makes a different precision-recall trade-off. If a different precision-recall tradeoff is desired, e.g. when selecting a particular class via a “quality cut”, downstream users can choose different classifier thresholds, or re-train our models using our Soft $F\beta$ loss for different values of β .

3.2 Unsupervised pretraining

As described in Section 2.5, we measure the generative performance of our model compared with the Gaussian Process of [Boone \(2019\)](#). We first analyzed the extreme values, those with KL divergence greater than 10. For our model, there were 464 such LC points in 10 examples. Almost all of these (98%) had flux values more than 500, so we assume they are most related to the known modeling deficiencies we mentioned in Section 2.5. For the GP model, there were 5968 such LC points in 862 examples. Unlike ATCAT, about 90% of the GP outliers had flux less than 500 (and the GP has no theoretical limitation on larger fluxes). From manual inspection, these generally corresponded to higher flux values and points at the beginning of the sequence.

We then filtered these points out, and focused on the common case, graphing its performance in Figure 6. Our model is significantly better; of these remaining 54,713 points, the mean of $\text{KL}(P||Q_{\text{GP}}) - \text{KL}(P||Q_{\text{ATCAT}})$ was 1.0. This outcome is consistent with our expectations, since the transformer is trained on 1.5M light curves, whereas the GP just has a few hyperparameters. However, [Boone \(2019\)](#) was the winner of the PLASTICC Kaggle competition, hence we believe their GP’s kernel selection and hyperparameters have been well-tuned for this task, and should be a reasonable baseline. While we used our LC-only model for fair comparison to the GP (which does not access metadata), it

Model	Architecture	# params	Dataset (# train/val/test)	# classes	LC-only	LC+meta
ATAT	transformer	1.9M	ELAsTiCC v1 (1.5M/365K/20K)	20	62.7 ± 0.4% F1	83.5 ± 0.6% F1
ORACLE	RNN		ELAsTiCC v2 (449K/225K/23K)	19	-	84% F1
SWINv2-based	vision transformer	29M	Exact same as ATAT	20	65.5 ± 0.28% F1	-
Ours	transformer	4.1M	Exact same as ATAT	20	71.79 ± 0.28% F1	89.75 ± 0.04% F1
Ours, unsup. pretrain	ensemble of transformers	4.1M / model	Unsupervised pretraining: 1.5M Supervised: 145K/365K/20K	20	67.44 ± 0.06% F1	87.07 ± 0.18% F1

Table 1. Results from several state-of-the-art models on ELAsTiCC. All $\pm$ values are simple `np.std()` calculations on the F1 metric per CV fold. We only ran our unsupervised pretraining experiment on the first 3 CV folds; others use all 5 folds. See text for details.

	RF (MD + Features)			ATAT (LC + MD + Features)			ATCAT (LC+metadata)		
Class	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
CART	59.2 ± 0.4	56.2 ± 0.6	57.6 ± 0.5	75.3 ± 2.5	40.0 ± 4.6	52.0 ± 3.3	78.8 ± 1.7	73.2 ± 1.0	75.8 ± 0.5
Iax	57.6 ± 0.5	55.9 ± 0.6	56.8 ± 0.5	59.8 ± 2.1	65.1 ± 5.4	62.2 ± 1.9	72.7 ± 1.0	80.9 ± 0.9	76.6 ± 0.9
91bg	75.2 ± 0.4	90.2 ± 0.2	82.0 ± 0.2	88.8 ± 2.2	92.5 ± 1.9	90.5 ± 0.6	93.5 ± 0.6	96.2 ± 0.3	94.9 ± 0.3
Ia	61.4 ± 0.4	76.7 ± 0.2	68.2 ± 0.3	76.3 ± 1.2	81.4 ± 1.7	78.8 ± 0.7	79.7 ± 0.5	86.9 ± 0.4	83.1 ± 0.3
Ib/c	58.0 ± 0.3	39.6 ± 0.4	47.1 ± 0.2	50.0 ± 3.8	65.8 ± 3.4	56.6 ± 1.2	69.6 ± 0.8	63.0 ± 0.9	66.1 ± 0.2
II	66.8 ± 0.6	42.7 ± 0.5	52.1 ± 0.5	63.9 ± 3.5	66.4 ± 2.8	65.0 ± 1.3	75.3 ± 0.3	74.6 ± 0.5	75.0 ± 0.3
SN-like/Other	59.0 ± 0.5	54.1 ± 0.8	56.5 ± 0.6	64.3 ± 2.2	60.5 ± 2.9	62.3 ± 1.5	70.6 ± 1.1	69.6 ± 1.1	70.1 ± 1.0
SLSN	90.3 ± 0.1	90.0 ± 0.1	90.2 ± 0.1	89.6 ± 0.9	95.4 ± 0.4	92.4 ± 0.4	96.1 ± 0.3	94.2 ± 0.4	95.1 ± 0.3
PISN	85.6 ± 0.1	96.7 ± 0.1	90.8 ± 0.0	95.9 ± 0.4	96.7 ± 0.9	96.3 ± 0.4	97.7 ± 0.2	98.4 ± 0.3	98.0 ± 0.2
TDE	83.2 ± 0.4	76.8 ± 0.3	79.9 ± 0.2	79.0 ± 4.9	92.5 ± 1.0	85.1 ± 2.6	93.0 ± 0.9	92.2 ± 0.4	92.6 ± 0.5
ILOT	76.3 ± 0.3	93.6 ± 0.2	84.1 ± 0.2	92.1 ± 0.9	84.0 ± 3.1	87.8 ± 1.3	92.7 ± 0.9	91.1 ± 0.3	91.9 ± 0.4
KN	86.8 ± 0.2	90.3 ± 0.1	88.5 ± 0.1	97.1 ± 0.4	77.1 ± 2.5	85.9 ± 1.4	95.2 ± 0.4	94.6 ± 0.6	94.9 ± 0.1
M-dwarf Flare	95.0 ± 0.3	79.4 ± 0.3	86.5 ± 0.3	99.1 ± 0.3	70.4 ± 1.9	82.3 ± 1.3	99.0 ± 0.3	89.5 ± 0.6	94.0 ± 0.4
uLens	96.9 ± 0.4	82.8 ± 0.2	89.3 ± 0.3	86.8 ± 1.7	95.6 ± 0.7	91.0 ± 0.7	93.4 ± 0.4	95.8 ± 0.7	94.6 ± 0.2
Dwarf Novae	78.5 ± 0.2	82.9 ± 0.3	80.6 ± 0.2	86.2 ± 1.8	92.0 ± 0.9	89.0 ± 0.9	91.7 ± 0.3	96.7 ± 0.2	94.1 ± 0.1
AGN	95.4 ± 0.4	99.9 ± 0.1	97.6 ± 0.2	99.7 ± 0.1	100.0 ± 0.0	99.8 ± 0.1	100.0 ± 0.1	100.0 ± 0.0	100.0 ± 0.0
Delta Scuti	90.8 ± 0.3	98.9 ± 0.0	94.7 ± 0.2	98.7 ± 0.3	99.5 ± 0.1	99.1 ± 0.1	99.3 ± 0.2	99.6 ± 0.1	99.4 ± 0.2
RR Lyrae	91.6 ± 0.4	98.9 ± 0.1	95.1 ± 0.2	99.5 ± 0.2	99.1 ± 0.2	99.3 ± 0.1	99.6 ± 0.2	99.5 ± 0.1	99.5 ± 0.1
Cepheid	92.6 ± 0.5	98.9 ± 0.1	95.6 ± 0.3	99.2 ± 0.3	99.5 ± 0.1	99.3 ± 0.1	99.5 ± 0.1	99.6 ± 0.2	99.6 ± 0.1
EB	93.5 ± 0.3	97.5 ± 0.1	95.5 ± 0.2	90.4 ± 1.7	99.6 ± 0.1	94.8 ± 0.9	98.4 ± 0.3	99.7 ± 0.1	99.0 ± 0.1

Table 2. Fine-class comparison. ATCAT has strong F1 scores on all fine classes, while occasionally making a different precision-recall tradeoff than the RF model or ATAT. $\pm$ values are from `np.std` over 5 CV folds.

Experiment	LC-only	LC+metadata
Baseline ^(a)	63.1 ± 0.6%	83.7 ± 0.1%
Increased regularization ^(b)	65.5 ± 0.1%	85.7 ± 0.4%
Ensemble only ^(c)	66.73 ± 0.15%	86.66 ± 0.13%
Pretrained ensemble ^(d)	67.44 ± 0.06%	87.07 ± 0.18%

Table 3. Unsupervised pretraining experiments. We remove labels from 90% of our normal training dataset, sampled uniformly at random. These experiments build on each other. Row (a) is our model with no modifications, trained on the remaining 10% of training data with labels. Row (b) increases regularization. Row (c) ensembles 10 sub-models. Row (d) adds unsupervised pretraining on the full train dataset (unsupervised pretraining does not use labels) and then does fine-tuning on the remaining labeled 10%. Values are Macro F1 scores, with $\pm$ from `np.std` over 3 CV folds. See text for details.

is interesting and potentially useful that we can condition our generation on the metadata.

We then look classification performance when removing labels from 90% of our training split, in Table 3. The simple step of increasing regularization is critical, as shown in the second row (settings in Section 2.4). Ensembling was also quite helpful; we hypothesize that it functions similar to regularization, and

ensures that no single example can disproportionately influence model output. We ensembled by splitting our dataset into 5 sub-folds (having already applied the main cross-validation split) and training 10 models, each with one sub-fold omitted. Our pretraining step provided less than 1% absolute F1 gain on both LC-only and LC+metadata, but at the scale of LSST we believe it is valuable. Furthermore, as argued in the introduction, we believe that this scenario will correspond to practical situations of having a vast quantity of unlabeled data (even more than 10:1), and pre-trained models should have better generalization and more useful last-layer embeddings.

3.3 Early detection

In Figure 7, we show early detection performance. When we added an early detection auxiliary loss described in Section 2.6 (dashed green line), both our LC-only and LC+metadata models outperform ATAT variants at all time scales, including ATAT’s MTA variant (masked temporal augmentation, a similar strategy to improve early detection performance, which uses data augmentation to truncate light curves at specific time cutoffs). Our model’s variance across CV folds was also much more tightly controlled, which is generally expected as model quality improves. However, we noticed slight drops in final accuracy when adding

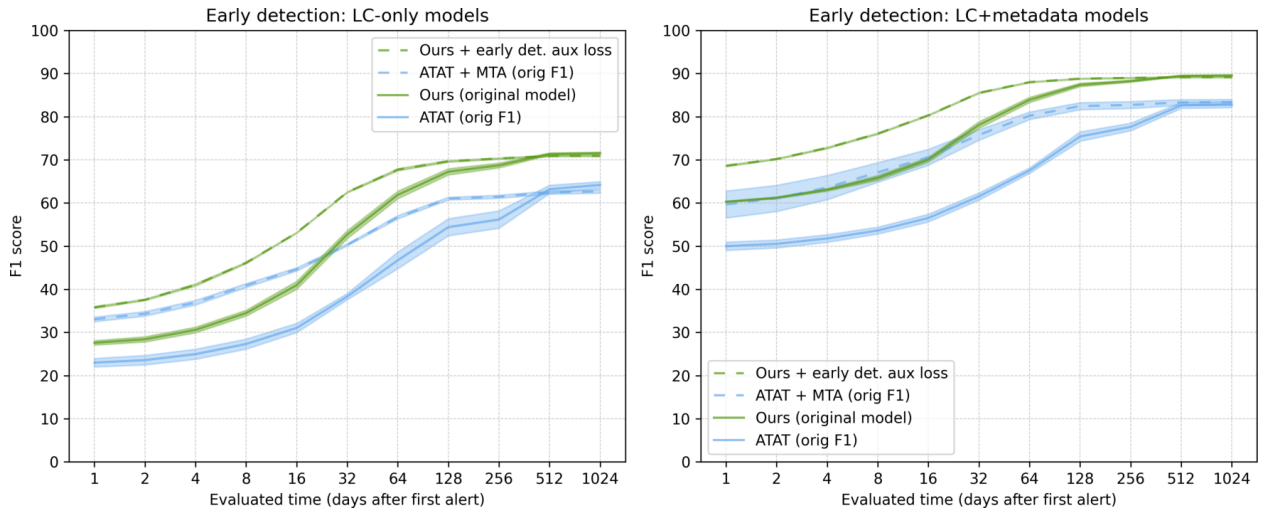


Figure 7. Early detection performance. The strong performance of ATCAT carries over to early detection setups. Variances are `np.std()` calculations over the 5 CV folds for both ATAT and ATCAT. See text for details.

the early detection loss, suggesting some remaining headroom in our model (see Section 4.6).

3.4 Calibration

Results for our calibration experiments for LC-only models can be found in Figure 8. Following standard practice in Guo et al. (2017), we used the score of the top-scoring class as the confidence, which both worked well in practice and works for models which are not trained to output good p_{unsure} scores. Models are free to generate confidence scores through any method though. After binning by model confidence, we adjusted the x-axis value to be the actual mean of binned samples; we believe this creates slightly more accurate charts when comparing to the red “ideal” line, although bins/points are not exactly evenly spaced. Results from LC+metadata models were similar, and are omitted for space reasons.

Our top-F1-scoring model is poorly calibrated. For example, when it outputs a probability score of “94%” for a class, it is only right 60% of the time. The training-time calibration loss is much better, and the bootstrap ensemble is extremely good, albeit being twice as slow to train.

Label smoothing (following Szegedy et al. 2015) produced model outputs that were closer to the calibration line, but in a way that very much reflected its mechanics: most top scores were now around 0.95 and few were highly confident. We believe this approach is less useful than the others; after all, one could take a baseline model and shift over all of its values, but it fails to further separate examples into ones the model is actually confident in. As we can see in our results, successful calibration techniques do still output high scores frequently for the ELAsTiCC dataset.

Focal loss with $\gamma = 0.5$ ended up producing a very well-calibrated model, but with a slightly greater cost to F1 (71.3%). It has the advantage of being a very simple technique, but higher values of γ end up with an under-confident model, necessitating tuning. We omit charts for space reasons.

3.5 Ablation studies

3.5.1 Input encoder

In Table 4 we examine varying input encoder changes. Our embedding scheme is the most significant accuracy gain in this work. Our default/final model is the last row, described in Section 2.2.1. In rows (a) and (b), we use our model with the ATAT embedder

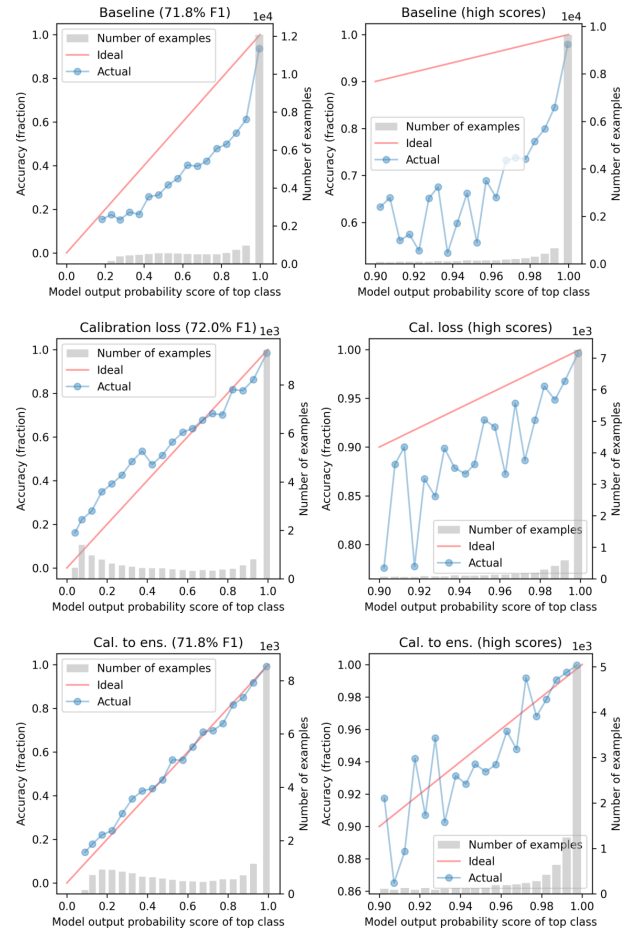


Figure 8. Calibration experiments. All models here are LC-only, trained on the first CV split; their performance on the test set at step 36000 is shown above. Each plot is a reliability diagram, which correlates the “confidence” of a model with its actual accuracy. Values above the red line indicate that the model is underconfident; values below indicate that the model is overconfident / “confidently wrong”. In this case of multi-class prediction, we interpret the “confidence” to be the probability of the top class (see text) and per-example “accuracy” to be 1 if the top predicted class (which can be less than 0.5) is the same as the actual class (ignoring the “unsure” output class) and 0 otherwise. The number of examples in each confidence bin is shown with grey bars. Each row is for a single model, for the entire range of output scores, and zoomed to the more confident ones.

Experiment	LC-only	LC+metadata
ATAT embedder ^(a)	61.53 ± 0.76%	79.79 ± 0.42%
ATAT + tanh inp. flux ^(b)	57.94 ± 0.13%	83.3 ± 0.04%
Posenc, no ch, no flux err, no scaling ^(c)	60.85 ± 0.12%	84.7 ± 0.12%
Posenc, no flux err, no scaling ^(d)	69.95 ± 0.07%	88.86 ± 0.09%
Posenc, no scaling ^(e)	71.13 ± 0.24%	89.37 ± 0.13%
Posenc ^(f)	71.73 ± 0.05%	89.5 ± 0.07%
ATCAT default ^(g) (rotary encoder, wavelengths)	71.87 ± 0.14%	89.86 ± 0.1%

Table 4. Ablations for embedding schemes. We break down our improvements over ATAT’s embedder into a variety of sub-steps. Our embedding scheme is the most significant accuracy gain in this work. All rows use the rest of our model/training regime, modifying only the embedder. Rows (a) and (b) use the ATAT embedder. Row (c) switches to a variant of Vaswani et al. (2017), although it chooses a set of time value scales appropriate to ELASrTiCC. In row (d) we add channel index information through an indexed linear embedding, in row (e) we add `flux_err`, in row (f) we add some dynamic range pre-scaling. In row (g) we replace the indexing channel approach from row (d) with our rotary encoder based approach, making the model more generally useful for non-ELASrTiCC data. Values are Macro F1 scores, with $\pm$ from `np.std` over 3 CV folds. See text for details.

(slightly re-implemented to accommodate appropriately shorter sequence lengths, as discussed in Section 3.6; we wrote tests to ensure equivalence). These results do not match the ATAT numbers exactly because we did not adjust other parts of the model; in particular, the LC+metadata models did not use the dynamic feature embedder, which includes many features based on light curve flux values. In row (b), we replacing calibrated `flux` with `tanh(flux/10)` (dividing by 10 to preserve dynamic range). We saw an improvement on LC+metadata but unexpected decrease on LC-only.

In row (c), we use an embedder which is closer to Vaswani et al. (2017), except with the important scaling of time values (see Section 2.2.1) to accommodate the uneven distribution we have, and tanh scaling of input flux at varying scales. Please see Cabrera-Vives et al. (2024) for a more literal implementation of Vaswani et al. (2017)’s embedder. In row (d), we add the channel information, in this case performing part of the linear transformation of time values indexed by channel. The ATAT experimentation with Vaswani et al. (2017)’s positional encoder did not include channel information—while this is not part its definition, it is an additional piece of data which is quite impactful. In row (e), we add the `flux_err` information, which provides a significant boost. In row (f), we add the scaling discussed in Section 2.2.1, which we implemented after comparing the dynamic range of activations when debugging our network; despite seeming arbitrary, it appears to be helpful. Finally, in row (g), we switch to a rotary encode using channel wavelengths. This provides a small accuracy boost for LC+metadata and less for LC-only. More importantly, our model more widely useable, since it is no longer tied to LSST color bands.¹

¹ As discussed in Methods, we do normalize the range by LSST bands, but this could be relaxed, and the model does not fail for wavelengths out of that range (which are created by our data augmentation process).

Experiment	Rate	LC-only	LC+metadata
Baseline No aug.		69.1%	88.8%
Flux scaling	0.2	69.2%	88.8%
Redshifting	0.1	70.1%	88.8%
Subsampling	0.25	70.7%	89.3%
Time shifting	0.2	70.4%	89.1%
ATCAT default		72.1%	89.5%
Random noise	0.15	69.7%	88.5%
Preset 2	*	71.4%	89.6%

Table 5. Ablations for data augmentation. We examine the effect of adding various data augmentations. Each row adds to the baseline; rows are not iteratively stacked, although the “ATCAT default” preset applies the 4 above augmentations, and “Preset 2” adds additional random noise to the ATCAT default. “Preset 2” was used for pretraining experiments, where we had less labeled data and wanted to add more regularization. Rates were lightly tuned. Values are Macro F1 scores from a single experiment on CV fold 0, so please beware only larger differences should be relevant. See text for details.

Experiment	LC-only	LC+metadata
ATCAT default ^(a)	71.87 ± 0.07%	89.66 ± 0.14%
Larger model ^(b)	71.69 ± 0.19%	89.85 ± 0.05%
Non-hybrid model ^(c)	71.86 ± 0.12%	89.29 ± 0.17%

Table 6. Ablations for non-embedding architecture choices. We explore two separate model variants. In row (b) we moderately increase several hyperparameters affecting model size. In row (c) we replace the local/hybrid attention with only global attention layers. Values are Macro F1 scores, with $\pm$ from `np.std` over 3 CV folds. See text for details.

3.5.2 Data augmentation

We implemented several types of data augmentation described in Section 2.1.2. Our results for these appear in Table 5. The subsampling data augmentation routine is the most effective. We hypothesize that this augmentation could be particularly effective because it always results in another valid light curve from that class.

3.5.3 Model architecture

In Table 6 we perform ablations on a few modeling choices (see section above for the more impactful input embedding choices). We experimented with a larger model with 512 embedding dimension (instead of 384), 8 layers (instead of 4), and 128 attention dimension (shared among 4 attention heads, so 32 per attention head). This was significantly slower and not much better, but may be worth trying on different datasets. Earlier experiments with local attention showed a small benefit, but in Table 6 we did not see much impact, and the “non-hybrid model” with only global attention layers worked just as well. However, if combined with a limit on the number of points which can attend to each other (say at most k points, even if we get an outlier cluster with more than k points within 1 day / 10 days), then local attention layers can be $O(kn)$ rather than $O(n^2)$, so this architecture may still be advantageous for speed even if it does not provide an accuracy boost.

Experiment	LC-only	LC+metadata
ATCAT default	71.79 $\pm$ 0.15%	89.91 $\pm$ 0.1%
Log-loss only	71.51 $\pm$ 0.14%	89.57 $\pm$ 0.04%
Soft F1 loss only	70.53 $\pm$ 0.11%	87.77 $\pm$ 0.16%
No “unsure” output	71.81 $\pm$ 0.04%	89.79 $\pm$ 0.09%
ADAM, no LR sched.	71.47 $\pm$ 0.11%	89.02 $\pm$ 0.1%

Table 7. Ablations for loss functions. We explore variants of losses and optimizer settings. Experiments should be interpreted as modifying the ATCAT default, and are not stacked on each other. Values are Macro F1 scores, with $\pm$ from np. std over 3 CV folds. See text for details.

Experiment	CPU/GPU	Rate [LCs/sec]
ATAT feature extraction	1c CPU	5.3
<i>above, scaled to approx. multicore</i>	*	303
ATAT model without features	A100	2105
ATAT model with features	A100	1206
Our model, LC only	40GB A100	41905
Our model, LC only	80GB A100	45568
Our model, LC + Meta	40GB A100	40283
Our model, LC + Meta	80GB A100	44273
ORACLE	H100	7692
Our model, LC + Meta	H100	68906
Our model, LC + Meta	RTX 4090	33153

Table 8. Inference performance. We measure the inference performance of ATCAT, our model. ATAT feature extraction was reported as a single-core number, so we have scaled it to approximate performance-per-watt of the A100 (see Appendix A.2 for details).

3.5.4 Losses and optimization

We also experimented with a few variants of loss functions and optimization methods in Table 7. In general, the log-loss is essential (i.e. only using Soft F1 loss underperforms significantly). Our default model, which adds Soft F1, does seem to provide a barely-significant performance boost. We also did an ablation, changing from our somewhat uncommon optimizer setting with $\beta_1 = 0$ (following ATAT) to a more vanilla configuration, although maintaining the $1e-4$ learning rate. This was significantly worse for the LC+metadata experiments.

3.6 Computational performance

We achieved generally favorable GPU performance over other state-of-the-art ELAsTiCC classification models. First, we used PyTorch compilation, which does a variety of optimizations and almost always results in significant speedups. Secondly, we matched the maximum sequence length of ELAsTiCC light curves exactly (243 points). In comparison to ATAT specifically, their model pads all channels to the same length, resulting in a sequence length of 384. Since the attention mechanism is $O(n^2)$, this means our model only needs 40% of the computation for this step. Of course, this advantage will disappear if we need to increase sequence length to support longer non-simulated light curves. Finally, we used `bfloat16` pervasively after input encoding. Half-precision floating point values are much faster on

modern GPUs; the A100 processes `float32` data at 19.5 Tflops, but `bfloat16` data at 312 Tflops (16x faster). The difference has only widened on more modern chips such as the H200, which processes `float32` data at 60 Tflops and `bfloat16` data at 1979 Tflops (30x faster).

Our results are in Table 8. We threw out the first 100 batches to ignore compilation and dataset prefetching. We report ATAT and ORACLE (Shah et al. 2025) results for batch size 2000, and our results at batch size 2048 for the A100 and 4096 for the H100. While larger batches sizes are important, around these optimal values, changes by a factor of 2 or 4 were not important. We also report RTX 4090 results (a consumer GPU 1/8 to 1/3 the cost of the A100), and recommend this setup.

Versus ATAT, the largest benefit is that our model does not require / benefit from CPU-bound feature extraction. ATAT’s best-performing model includes dynamic features, which need to be extracted every time for optimal performance. In other words, unlike most metadata, they cannot be cached for one object and re-used as new LC measurements come in. After applying a reasonable scaling to ATAT’s single-core results (see Appendix), we find that our model is approximately 174x faster. ATAT also has a model variant that skips these dynamic features for a modest accuracy penalty; compared to this setup, our model is 35x faster. We ran head-to-head comparisons with ORACLE as well, a RNN-based network which is focused more heavily on high throughput. On the H100, our model is 9x faster. However, RNNs’ sequential nature generally means they perform worse on GPUs and relatively better on CPUs, so we expect that ORACLE has better CPU performance. We haven’t had time to investigate CPU performance of ATCAT; please see Section 4.7.

3.6.1 Training performance

We did light optimization work for training speed. This was helpful for development speed and costs. Our data loader, with all augmentations, can load light curves at 192 batches/sec on a AMD Ryzen 9 7900X (batch size 256, so 49,000 LC/s). We achieved this while using a Polars dataframe-based implementation, which allowed for expressive runtime transformations. The trainer usually runs around 35 steps/s on a RTX 4090 (batch size 256, so 9,000 LC/s). Full training runs usually take around 45 minutes on an RTX 4090 and cost around \$0.35 on our cloud host; by contrast, Shah et al. (2025) reports 12 hour training times and 4 hour evaluation times on an H100 GPU, which is considerably more expensive.

4 DISCUSSION AND FURTHER WORK

In this section, we discuss the value of ATCAT, provide confusion matrices, discuss some possible approaches for anomaly detection, and share our top ideas for improvement.

4.1 Summary and value of contributions

In this work, we have shown that our model ATCAT (@🐱) significantly advances the state of the art in ELAsTiCC classification accuracy. We have also shown that it can be tuned for label-efficient training and generative modeling, early classification, and calibrated classifications.

As real LSST data becomes available, we hope that our recipes for unsupervised pretraining can be leveraged to make use of vast quantities of unlabeled data, and to train effective classifiers when labeled data is harder to come by. For example, for some tasks it may be desirable to train models based on expert-labeled classifications, rather than simulator outputs, and producing a

high-quality labeled dataset as large as ELAsTiCC would likely be infeasible. Employing all of our techniques in Section 3.2 should be a good first step. Domain adaptation is a challenging problem however, and much more work will be needed, as we discuss later.

For early detection, we demonstrated state of the art results with meaningful improvements. This performance boost will result in the ability to follow up on more objects of interest, or reduce “wasted” telescope time following up on less interesting objects. These follow-up observations usually use a more narrow field of view telescope, and/or telescopes with spectroscopic tools, which are in high demand for a variety of science goals.

For cosmological and stellar population studies, our work on accuracy-calibrated versions of ATCAT should be useful, since selecting by confidence score of a calibrated model allows selection by expected model precision. Our improvements in accuracy should mean that if researchers use our models for quality cuts, they will be able to select more objects at the same level of precision. However, distributional shifts remain a problem, and calibration does not mitigate classifier bias, as we discuss later.

Finally, our basic work on inference performance should be quite impactful. We hope that ATCAT can be run regularly as part of LSST alerting pipelines, for larger surveys, and anything in-between. ATCAT’s inference throughput implies a theoretical lower bound cost of \$3 per billion objects (33,000 LC/s on a RTX 4090 cloud worker at \$0.35/hour), though full pipeline deployment will undoubtedly be more expensive. Faster models also allow for faster experimentation cycles. Lastly, analysis of LSST data is a task carried out by scientists throughout the world, some of whom do not have supercomputer access. We have managed to both significantly improve upon state-of-the-art accuracy, and create a model that is faster and less expensive to run.

4.2 Confusion matrix

We provide a confusion matrix for LC-only and LC+metadata variants of ATCAT in Figure 9, for comparison with other papers, and understanding which specific classes are confused (beyond per-class F1 scores in Table 2). We grouped the confusion matrix by coarse class, which helps clarify errors between fine classes and between coarse classes, and have published this code for any subsequent work to utilize if they wish. We refer readers to [Shah et al. \(2025\)](#) for a good analysis / discussion of some common confusions. Although ATCAT has increased accuracy, its pairwise confusions remain similar to other works. Finally, despite all of the variable stars (bottom right of the confusion matrix) having high scores, we noticed that some of the misclassified ones could be period folded and identified as their true class. Therefore, for astronomers doing variable star studies, if maximum accuracy is critical, we recommend combining ATCAT with period folding logic (possibly adding the detected period(s) as input features).

4.3 Preliminary work: Anomaly detection

We perform some preliminary investigation into the ability to use ATCAT embeddings to detect novel types of objects. Work into anomaly detection has traditionally used isolation forests, and custom algorithms have also been applied ([Ishida et al. 2019](#)). Our experimental setup consists of training ATCAT with one or more classes of objects held out, and then investigating the embeddings of those held-out object classes. This is only to demonstrate feasibility; it is not representative of situations where there will be many observations away differing from the training set (especially a synthetic one like ELAsTiCC), and one has to choose among hundreds of outliers / outlier clusters, some of which are

genuine rare / novel events, and some of which are cases where the classifier was unsure / incorrect.

We tried two main approaches. The first consisted of fitting Gaussian mixture models (GMMs) to embeddings, and looking at the likelihood of held-out versus in-distribution examples. The intuition behind this approach is that our training data should lie in a subspace of the embedding, and novel objects should lie in another, and therefore have relatively low probability in relation to the training data. The results are shown in Figure 10. Easier classes like AGNs may be able to be picked out, but harder classes like CARTs are not distinguishable.

Our second approach was to train an ATCAT variant to have a “possibly novel class” output logit. (There are now 2 extra outputs: one for “novel” and another for “unsure”.) For each batch (256 LCs), we computed the mean and standard deviation of last layer embeddings. These statistics were computed element-wise (so there are 384 of them), over batches. We then generated random activations matching these means and standard deviations, and trained the final classifier network to output “novel” for them. For the actual LCs, we trained towards a KL loss with 91% weight on the true label and 9% on the “unsure” logit. We then added another auxiliary loss towards minimizing log-likelihood on the “novel” output class for actual LCs, clamping at -10 log likelihood. These losses are intended to make the model output “unsure” on points near its input domain, and “novel” only on points farther from its input domain.

This approach was more successful, as shown in Figure 11. AGNs, our easiest class, were detectable with near-100% precision at a good fraction of recall. For CARTs, our hardest-to-classify class (see Figure 9), we only got near-100% precision at quite low precision. In a hypothetical situation where we wanted to detect a new type of object like CARTs, this means that there would have to be a lot of them, but it might be possible, especially if we proceeded by studying the objects with highest “novel” score first. However, this is a preliminary result. The fact that some CART scores *decreased* in “is novel” score when we increased the aux loss suggests that this loss is not very robust. One possible direction for future work is trying to use our model’s generative capabilities, generating improbable but not impossible light curves via temperature sampling (see e.g. [Hinton et al. 2015](#) for temperature sampling).

4.4 Discussion of model bias

We now briefly discuss model bias. The term “bias” has many definitions; for example, in classical statistics, we might want to estimate the mean of a distribution, and ensure that our estimator on average (in expectation) returns the true mean. But this typically doesn’t make sense in classification, where we think of well-tuned models hedging between classes when they are uncertain, and never returning probabilities scores for a class that are negative or greater than 1.

One type of bias we might be interested in is among the distribution of classifier outputs themselves. For example, suppose we want to estimate how many supernovae are Ib/c vs. Iax. If we force the classifier to classify each object (even if it is unsure), then for our class-balanced test distribution, our calibrated LC-only model will find that there are $\approx 1.7x$ more Iax objects. Clearly, calibrating scores to accuracy still results in class imbalances. Filtering to those with $\geq 80\%$ confidence score, our sample has $\approx 1.67x$ more Iax objects. The more accurate LC+metadata model will find $\approx 1.3x$ more Iax objects, and the imbalance gets larger with the $\geq 80\%$ confidence score cutoff, now featuring $\approx 1.6x$ more Iax objects. The effect is loosely correlated with classifier score: classes with very good precision/recall will have counts near their

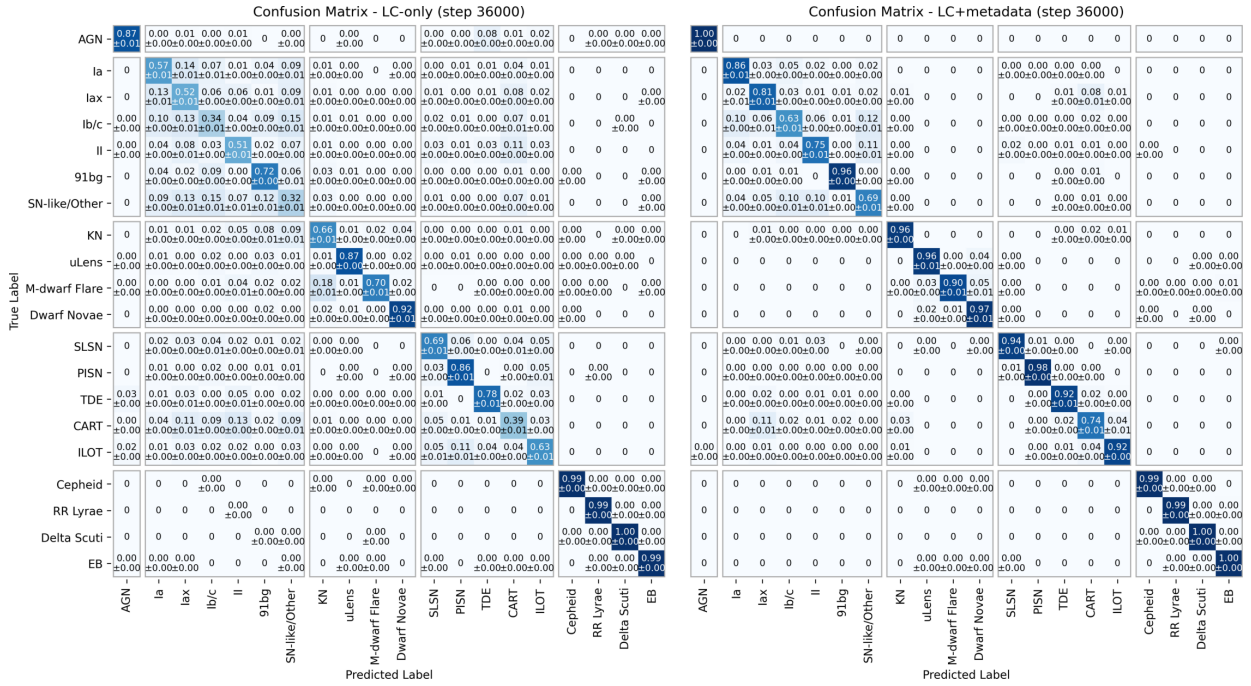


Figure 9. Confusion matrices. For each fold, we first accumulated counts indexed by true label and predicted label. Then we normalized by the true label, so each row sums to one. Finally, we take the mean and `np.std` over all of these matrices and visualize the result. We have segmented the labels into classes similar to [Shah et al. \(2025\)](#), so values within blocks on the diagonal can be interpreted as confusion between fine classes of the same coarse class, whereas blocks off the diagonal can be interpreted as confusion between coarse classes (for some purposes, this may be a more serious error).

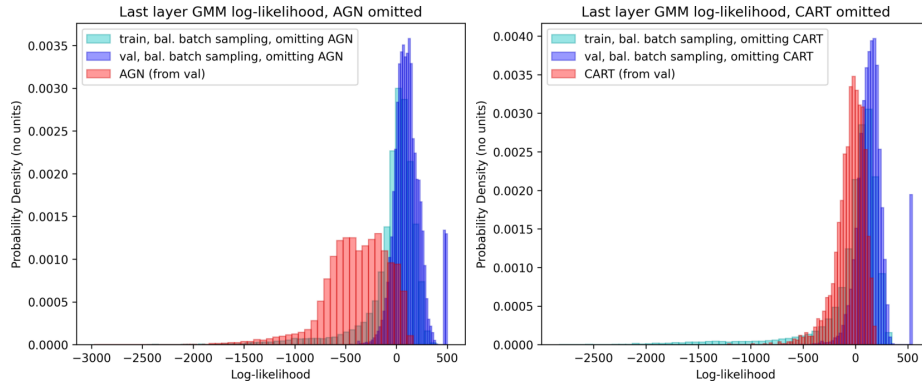


Figure 10. GMM clustering for anomaly detection (preliminary work). We apply train a GMM on the last layer embeddings and see if it can detect outliers. The GMM was trained on a balanced-batch version of our validation set (the ATCAT model was trained on the training set, minus the class in question). What we’re looking for is separation of our “novel” (held out) class from the rest of the data; see text.

true class counts, but other classes may be above and below, and the distance isn’t strictly correlated with F1 score.

Another type of bias is the conferred bias when making quality cuts using our model. It is very common to study a property of objects by making quality cuts, see [Gaia Collaboration et al. \(2018\)](#) for example. Let’s call this property $\mathfrak{A}$; it could be numeric (e.g. estimated mass) or binary (e.g. is the mass less than some threshold). The key question is: if we select objects using the classifier outputs (say everything matching class c with confidence s), how will $\mathfrak{A}$ change? We propose that the risk depends on a few factors: the base accuracy of the classifier, how $\mathfrak{A}$ appears in mis-classified objects, and $\mathfrak{A}$ ’s correlation with classification scores. At the limit where the classifier is very good, then our quality cut will be equal to the true sample, and the other concerns are not relevant. Otherwise, we will be choosing a model score threshold s with a certain precision-recall tradeoff (often visualized with a precision-recall curve; we recommend Chapter 8 of [Manning et al. 2008](#) to unfamiliar readers). Higher recall selections (low value of s) will select many objects of different

classes, so the result of our study will mostly depend on how $\mathfrak{A}$ manifests in mis-classified objects. If $\mathfrak{A}$ is extremely different in mis-classified objects, then these outliers can be filtered out; conversely, if $\mathfrak{A}$ has the same distribution in mis-classified objects, then the result of our study won’t change. So the most adversarial case lies in the middle. Higher precision selections (high value of s), by contrast, will depend more on the correlation of $\mathfrak{A}$ with the classification score. Particularly, if some settings of $\mathfrak{A}$ make it easier or harder to classify, then our high precision sample will be skewed.

In Appendix A.3, we studied this empirically on two simulator parameters, `LOGMASS_TRUE` and `LOG_SFR` (removed from our datasets by default). We tried to find more interesting/varying parameters, but were unable to find ones that had terribly pathological behavior, in part because these parameters are only present on some classes of objects. We studied the property over different precision/recall tradeoffs, since high-precision and high-recall selections may be affected through different phenomena.

Causes and ameliorations of bias also vary. One serious issue

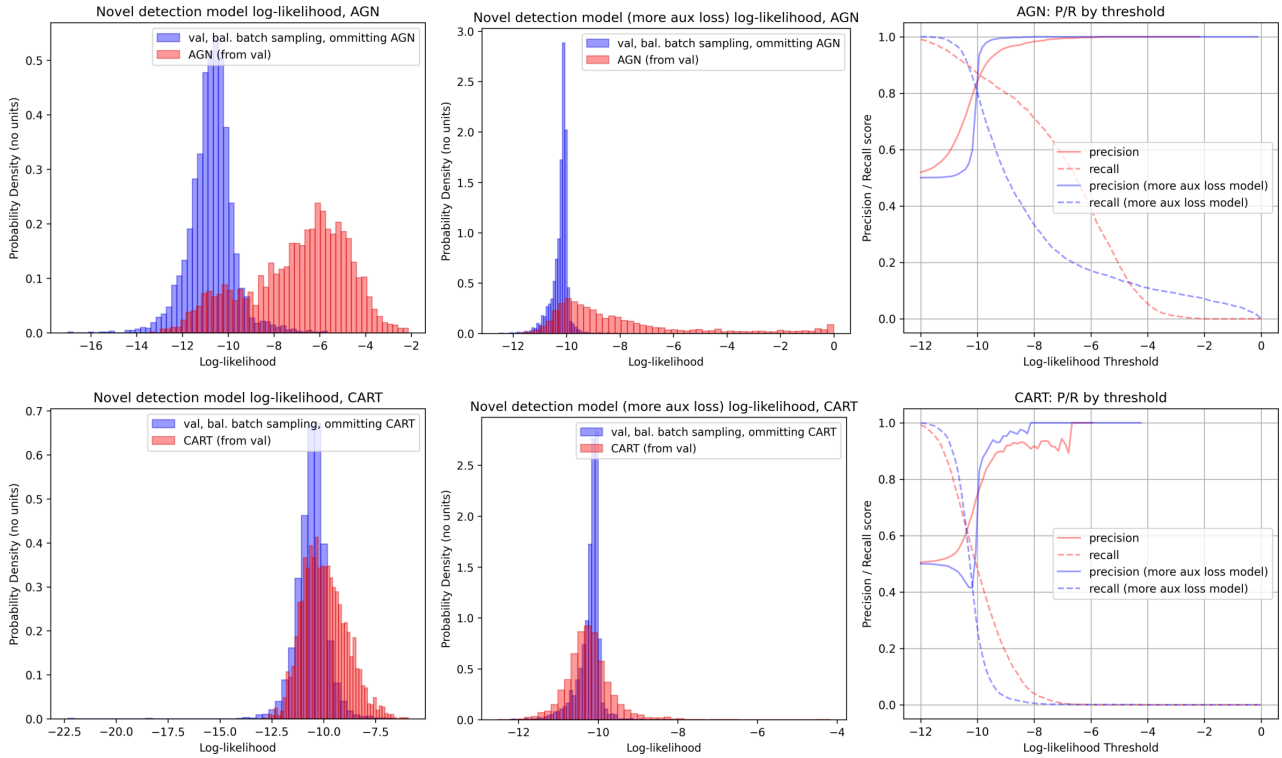


Figure 11. Model variants for anomaly detection (preliminary work). We train a variant of ATCAT that has an extra “novel” output class, and train the final classifier to output this class when last-layer embeddings are anomalous. The left and center columns show novel detection logit outputs. The first row shows results for AGNs (easy), the second for CARTs (harder). The left column has a base weighting of the auxiliary loss, and the middle column weights this loss even more, and further shrinks random vector lengths to match the actual data standard deviations. In these setups, train and test distributions (shown in Figure 10) were very similar to val, so we omitted them for readability. What we’re looking for is separation of our “novel” (held out) class from the rest of the data. The right column show precision/recall by threshold, if we were to retrieve all examples greater than a certain score threshold.

is domain adaptation. When a model is trained on one domain and evaluated on another, then this can not only harm its overall accuracy, but lead to objects of one type systemically being classified as another. Richards et al. (2011) studies this effect on variable star classification on the OGLE (Optical Gravitational Lensing Experiment) dataset, and finds that active learning is quite effective.

Future strategies for amelioration may also draw inspiration from the large body of ML fairness work for analysis and potential solutions to the problem (see for example D’Amour et al. 2020; Ayres 2002; Corbett-Davies et al. 2017). Perhaps surprisingly, the statistical structure of the latter type of bias we discuss appears to be similar to certain fairness problems. ML fairness deals with problems like correlation of skin tone (analogous to our “property $\mathfrak{A}$ ”, often not labeled in the original training data) and skin condition classifier performance (analogous to our classifier output).

To conclude, we strongly encourage practitioners to find a reasonable statistical setup for their problem, instead of simply using our classifier outputs as priors, even after calibration. We should be specific about what calibration tries to achieve,

$$P_{X,Y \sim \mathcal{D}} \left(Y = \underset{c}{\operatorname{argmax}} f_c(X) \mid \max_c f_c(X) \approx s \right) \approx s$$

e.g. if the classifier confidence (top output score) is around 90% ($s = 0.9$), then for a well-calibrated model on a representative dataset, 90% of these examples will have the correct label. Calibration does not provide a guarantee about the balance of top classifications, or how likely the classifier is to be confident of examples in a particular class. It is, however, useful for estimating the precision of a sample, for situations when we are trying to manage the model’s precision-recall tradeoff.

4.5 Future work: Generative possibilities

Since our model has generative capabilities (it predicts the next flux given the time, channel wavelength, and flux_err) and can classify such LCs, it would be valuable to look at expected informational gain. One way of defining this could be

$$E_{x_{t+1}} [\text{KL} (q(Y|x_{t+1}, x_t, \dots) || q(Y|x_t, \dots))]$$

following Li et al. (2024). This could be used to evaluate proposed sequences of observations; for example, a telescope path and color filter selection might specify time and channel wavelength values, and then flux_err values could be drawn from a simple distribution (e.g. sampling randomly from previous flux errors). Then, outer expectation could be approximated by sampling next flux values from the model’s generative output. The inner KL term is the discrete multinomial KL divergence; before adding the new point, the classifier gives us a vector of probabilities over classes ($q(Y|x_t, \dots)$), and we compare that to the probabilities over classes after the new point has been added ($q(Y|x_{t+1}, x_t, \dots)$). However, there are probably many cases where one has technically gained information, but the distinctions aren’t as interesting, for example one experimenter might not care about distinguishing RR-Lyrae vs. Delta Scuti’s, but might care a lot about CARTs vs. Type-II supernovae. In that case, collapsing classes, or choosing another metric than KL altogether would be most valuable.

4.6 Future work: Further modeling improvements

Combining ATCAT’s architecture with other techniques should also be a fruitful avenue of exploration.

Coarse-grained classifications may be useful, especially for early detection. These can both help to focus on classes / class

boundaries of more specific interest (for example, focusing on rare transients such as KN, TDE, etc. and excluding more common ones such as Type Ia and variable stars), and also to achieve higher accuracy via hierarchical losses, as demonstrated by ORACLE (Shah et al. 2025).

Applying ATCAT to other surveys, and training a cross-survey variant would also be interesting and potentially valuable. We would specifically like to combine ZTF and LSST data. We hope our work making the model work on only channel wavelengths instead of integer channel indices (which required a change to the embedder) will be helpful. A cross-survey model could also be combined with expected informational gain metrics (see Section 4.5) to prioritize follow-up observations with different telescopes. We would again like to stress the value of better dataset formats and standardization (Section 4.8); in this case it is also critical to have light curves assembled reflecting the same source observed in multiple channel wavelengths, since it is a prohibitive amount of work to simultaneously do this and build cross-survey models.

Finally, low-level improvements in the ATCAT architecture may be valuable. Despite our advances in modeling accuracy, we believe we have not removed all of the headroom. For LC encoding, Su et al. (2021) also found that applying the rotary encoder in the attention mechanism was better. We would also like to explore the effects of repeated observations; while it is not a-priori bad that our model is sensitive to them (they may reflect increased confidence, for example), its sensitivity may differ from convolutional approaches. In Section 3.5.3 we saw slightly lower LC-only performance on larger models, which indicates sub-optimal regularization, initialization, or architecture choices. For early detection, we found that better settings for early detection resulted in a slight loss of accuracy on classifications after 1024 days since the initial alert. A “composite” model which switches between our base models and early detection models (based on the number of points) would be trivially better, indicating some modeling, loss, or training headroom. We also believe that more advanced schemes for weighting the loss could be useful, if it could be deduced at which LC points the model should be able to make a classification, and which ones are just adding noise. The early detection loss model is also a bit slower to train, possibly due to larger all-reduce steps in the gradient calculation.

4.7 Future work: Further performance improvements

Despite our competitive showing in training and inference performance, we believe significant headroom still exists. We suggest investigating further sequence length reductions primarily. This could come as simply truncating / subsampling long sequences, or doing a more sophisticated sequence packing scheme. Our current masking scheme is quite wasteful. The length of ELAsTiCC sequences are quite imbalanced; with our / ATAT’s splits, in the training set, only 17.7% of sequence elements are *not* masked on average, and in the test set, which has even sampling of classes, 26.0% were not masked. So this is a huge potential speedup for both training and inference. Unfortunately, sequence packing does usually have a medium-high code complexity cost to logic in datasets, models, and loss functions. We also suggest experimenting with a hard restriction on number of LC points (say, k) local attention layers can attend to, changing the attention time complexity from $O(n^2)$ to $O(kn)$ for those layers. We use the `flex_attention` library (Dong et al. 2024), which claims to optimize for this case (called “sliding window attention”).

Furthermore, if inference performance of ensembled models is important, then it should be investigated whether the results can be distilled into a single model (e.g. using KL divergence towards silver labels). We also did preliminary experiments with stochastic

weights averaging, but got inferior results to ensembling; further investigation may be reasonable.

ATCAT is a small enough model that it should be possible to make it run reasonably on the CPU. Additionally, several CPUs, including popular laptop CPUs, now include a small integrated GPU or NPU (neural processing unit), which could be leveraged using PyTorch backends MPS for Apple’s M-series chips, ROCm for AMD’s APUs, or OpenCL for several platforms.

4.8 Future work: Better dataset standardization

As mentioned in Section 3.1, it is unfortunate that there are many variants of ELAsTiCC preprocessing, which makes it difficult to compare models. Both for ELAsTiCC and other surveys, several strategies may help facilitate these comparisons. We suggest the following,

- Using Parquet files,² where each row is a LC. In the course of this work, we experimented with Dask, Lance, and Polars, and **highly recommend use of Polars and Parquet**. We suggest sharding and zstandard compression; with the right shard size, this provides good compression and multi-threaded loading. Metadata become “normal” columns, and LC points can become list-type columns.
- Ensuring all fields are well-documented, including their units, and methodologies and rationale behind upstream processing (such as mean field subtraction). If it is expected that many practitioners will filter by a particular bitmask³ / flag, then clean, idiomatic Python code for that should be provided.
- Making datasets usable with minimal preprocessing for basic tasks. When there are good defaults (such as filtering by flags), we suggest applying those to the main dataset, and separately sharing dataset variants without these defaults. If issues arise, such as SNIDs being duplicated between classes, then there should be a mechanism for fixing this at the source, perhaps using minor version numbers (e.g. “v3.1”).
- Declaring a standard schema for expressing training / validation / test splits, and collapsing of fine classes (labels). We suggest Parquet tables with SNIDs for splits (a balance of being explicit and efficient), and either a Parquet table, or clean, idiomatic Python code expressing fine classes collapsing logic. Different astronomers will care about modeling different aspects, so collapsing classes is natural and shouldn’t be dictated by the original dataset providers, but schemas / standardization would make it easier to directly compare models.
- Having clear licensing terms, specifying whether / how derivative datasets can be distributed.

We are hesitant to provide these solutions ourselves, to avoid proliferating standards.⁴ We hope that the above suggestions can be received as constructive criticism for anyone releasing datasets. Our criticism is not directed at ELAsTiCC as much as the state of ELAsTiCC classification ~2 years after its release. Clearly, our work would not have been possible without the work of the ELAsTiCC team, who assembled a dataset from a large variety of simulators; their work has realized immense scientific value for the LSST, and will continue to do so.

² Any Parquet-compatible implementation can be used, although as of writing Pandas notably lacks support for lists, and we suggest it not be prioritized given Polars’ vastly superior performance and more principled API.

³ Parquet supports boolean (single bit) or enum values, which may be cleaner than traditional integer bitmasks. Flags can also be combined into a struct, for grouping or reducing the number of top-level columns.

⁴ <https://xkcd.com/927/>

5 ACKNOWLEDGMENTS

I would like to thank the ATAT team for making their code and results publicly available, including all necessary components to reproduce the results in their paper. I would also like to thank Josh Bloom for many illuminating discussions, and inviting me to join discussions with his research group.

REFERENCES

- Ansari A. F., et al., 2024, arXiv preprint arXiv:2403.07815v3
- Ayres I., 2002, *Justice Research and Policy*, 4, 131
- Bailey S., Aragon C., Romano R., Thomas R. C., Weaver B. A., Wong D., 2007, arXiv preprint arXiv:0705.0493
- Bender E. M., Gebru T., McMillan-Major A., Shmitchell S., 2021, in Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. FAccT '21. Association for Computing Machinery, New York, NY, USA, p. 610–623, doi:10.1145/3442188.3445922, <https://doi.org/10.1145/3442188.3445922>
- Bloom J. S., et al., 2012, *Publications of the Astronomical Society of the Pacific*, 124, 1175
- Boone K., 2019, arXiv preprint arXiv:1907.04690v2
- Boone K., 2021, *The Astronomical Journal*, 162, 275
- Cabrera-Vives G., et al., 2024, arXiv preprint arXiv:2405.03078v2
- Carrasco-Davis R., et al., 2019, *Publications of the Astronomical Society of the Pacific*, 131, 108006
- Charnock T., Moss A., 2017, *Astrophysical Journal Letters*, 837, L28
- Corbett-Davies S., Pierson E., Feller A., Goel S., Huq A., 2017, in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '17. Association for Computing Machinery, New York, NY, USA, p. 797–806, doi:10.1145/3097983.3098095, <https://doi.org/10.1145/3097983.3098095>
- D'Amour A., et al., 2020, arXiv preprint arXiv:2011.03395v2
- Dong J., Feng B., Guessous D., Liang Y., He H., 2024, arXiv preprint arXiv:2412.05496
- Donoso-Oliva C., Becker I., Protopapas P., Cabrera-Vives G., Vishnu M., Vardhan H., 2023, *Astronomy & Astrophysics*, 670, A54
- Donoso-Oliva C., Becker I., Protopapas P., Cabrera-Vives G., Cádiz-Leyton M., Moreno-Cartagena D., 2025, arXiv preprint arXiv:2502.02717
- Durkan C., Bekasov A., Murray I., Papamakarios G., 2020, nflows: normalizing flows in PyTorch, doi:10.5281/zenodo.4296287, <https://doi.org/10.5281/zenodo.4296287>
- Erhan D., Bengio Y., Courville A., Manzagol P.-A., Vincent P., Bengio S., 2010, *J. Mach. Learn. Res.*, 11, 625–660
- Fraga B. M. O., et al., 2024, arXiv preprint arXiv:2404.08798v2
- Gagliano A., Contardo G., Foreman-Mackey D., Malz A. I., Aleo P. D., 2023, *The Astrophysical Journal*, 954, 6
- Gaia Collaboration et al., 2018, *A&A*, 616, A10
- Goyal P., et al., 2021, arXiv preprint arXiv:2103.01988v2
- Guo C., Pleiss G., Sun Y., Weinberger K. Q., 2017, arXiv preprint arXiv:1706.04599v2
- Gupta R., Muthukrishna D., Audenaert J., 2025b, arXiv preprint arXiv:2510.12958
- Gupta R., Muthukrishna D., Rehemtulla N., Shah V., 2025a, arXiv preprint arXiv:2502.18558v2
- Gómez C., Neira M., Hoyos M. H., Arbeláez P., Forero-Romero J. E., 2020, arXiv preprint arXiv:2004.13877v2
- Hinton G., Vinyals O., Dean J., 2015, arXiv preprint arXiv:1503.02531
- Ishida E. E. O., et al., 2019, arXiv preprint arXiv:1909.13260v2
- Jamal S., Bloom J. S., 2020, *Astrophysical Journal Supplement Series*, 250, 30
- Koh P. W., et al., 2020, arXiv preprint arXiv:2012.07421v3
- LSST Science Collaborations 2009, The LSST Science Book. LSST Corporation, <http://www.lsst.org>
- Lee H.-P., Yang Y.-J., von Davier T. S., Forlizzi J., Das S., 2023, arXiv preprint arXiv:2310.07879v2
- Li F., Baptista R., Marzouk Y., 2024, arXiv preprint arXiv:2411.08390v2
- Li W., Chen H.-Y., Lin Q., Rehemtulla N., Shah V. G., Wu D., Miller A. A., Liu H., 2025, arXiv preprint arXiv:2510.06200
- Lin T.-Y., Goyal P., Girshick R., He K., Dollár P., 2017, arXiv preprint arXiv:1708.02002v2
- Liu Z., Lin Y., Cao Y., Hu H., Wei Y., Zhang Z., Lin S., Guo B., 2021, arXiv preprint arXiv:2103.14030v2
- Lochner M., McEwen J. D., Peiris H. V., Lahav O., Winter M. K., 2016, *The Astrophysical Journal Supplement Series*, 225, 31
- Long J. P., El Karoui N., Rice J. A., Richards J. W., Bloom J. S., 2012, *Publications of the Astronomical Society of the Pacific*, 124, 280
- Malanchev K., 2023, in American Astronomical Society Meeting Abstracts #241. p. 117.03
- Manning C. D., Raghavan P., Schütze H., 2008, An Introduction to Information Retrieval. Cambridge University Press, Cambridge, England, <https://www-nlp.stanford.edu/IR-book/>
- Mohri M., Medina A. M., 2012, arXiv preprint arXiv:1205.4343v2
- Moreno-Cartagena D., Protopapas P., Cabrera-Vives G., Cádiz-Leyton M., Becker I., Donoso-Oliva C., 2025, Leveraging Pre-Trained Visual Transformers for Multi-Band Photometric Light Curve Classification (arXiv:2502.20479), <https://arxiv.org/abs/2502.20479>
- Muthukrishna D., Narayan G., Mandel K. S., Biswas R., Hložek R., 2019, arXiv preprint arXiv:1904.00014v2
- Möller A., de Boissière T., 2019, *Monthly Notices of the Royal Astronomical Society*, 491, 4277
- Naul B., Bloom J. S., Pérez F., van der Walt S., 2018, *Nature Astronomy*, 2, 151
- Opitz J., Burst S., 2019, arXiv preprint arXiv:1911.03347v3
- Perrigo B., 2023, Exclusive: OpenAI Used Kenyan Workers on Less Than \$2 Per Hour to Make ChatGPT Less Toxic, TIME Magazine, <https://time.com/6247678/openai-chatgpt-kenya-workers/>
- Pimentel Ó., Estévez P. A., Förster F., 2023, *Astronomical Journal*, 165, 18
- Qu H., Sako M., Möller A., Doux C., 2021, arXiv preprint arXiv:2106.04370
- Radford A., et al., 2021, CoRR, abs/2103.00020
- Raji I. D., Kumar I. E., Horowitz A., Selbst A. D., 2022, arXiv preprint arXiv:2206.09511v2
- Rehemtulla N., et al., 2024, arXiv preprint arXiv:2401.15167
- Richards J. W., et al., 2011, *The Astrophysical Journal*, 744, 192
- Rizhko M., Bloom J. S., 2024, arXiv preprint arXiv:2411.08842
- Shah V. G., Gagliano A., Malanchev K., Narayan G., Collaboration T. L. D. E. S., 2025, arXiv preprint arXiv:2501.01496
- Su J., Lu Y., Pan S., Murtadha A., Wen B., Liu Y., 2021, arXiv preprint arXiv:2104.09864v5
- Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z., 2015, arXiv preprint arXiv:1512.00567v3
- Tan A., Protopapas P., Cádiz-Leyton M., Cabrera-Vives G., Donoso-Oliva C., Becker I., 2025, arXiv preprint arXiv:2509.24134
- Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I., 2017, arXiv preprint arXiv:1706.03762v4
- Vidal E. P., Gagliano A. T., Cuesta-Lazaro C., 2025, arXiv preprint arXiv:2510.14202
- Villar V. A., et al., 2019, *The Astrophysical Journal*, 884, 83
- Woo G., Liu C., Kumar A., Xiong C., Savarese S., Sahoo D., 2024, arXiv preprint arXiv:2402.02592v2
- Zhang G., Helfer T., Gagliano A. T., Mishra-Sharma S., Villar V. A., 2024, arXiv preprint arXiv:2408.16829
- Željko Ivezić et al., 2008, arXiv preprint arXiv:0805.2366v5

A APPENDIX

A.1 Class exemplars and “mis-exemplars”

We provide several example light curves, to help make our classification problem concrete. In Figure 12 we show class exemplars, or examples in each class where our LC-only model gave the example a high score. In Figure 13 we show “mis-exemplars”, examples where our model gave a very low score to the true class. And in Figure 14 we specifically examine some pairs of classes which had high confusion scores in the confusion matrices (Figure 9), visualizing exemplars from each class on the left and right, and then confusions in the middle two columns (examples from

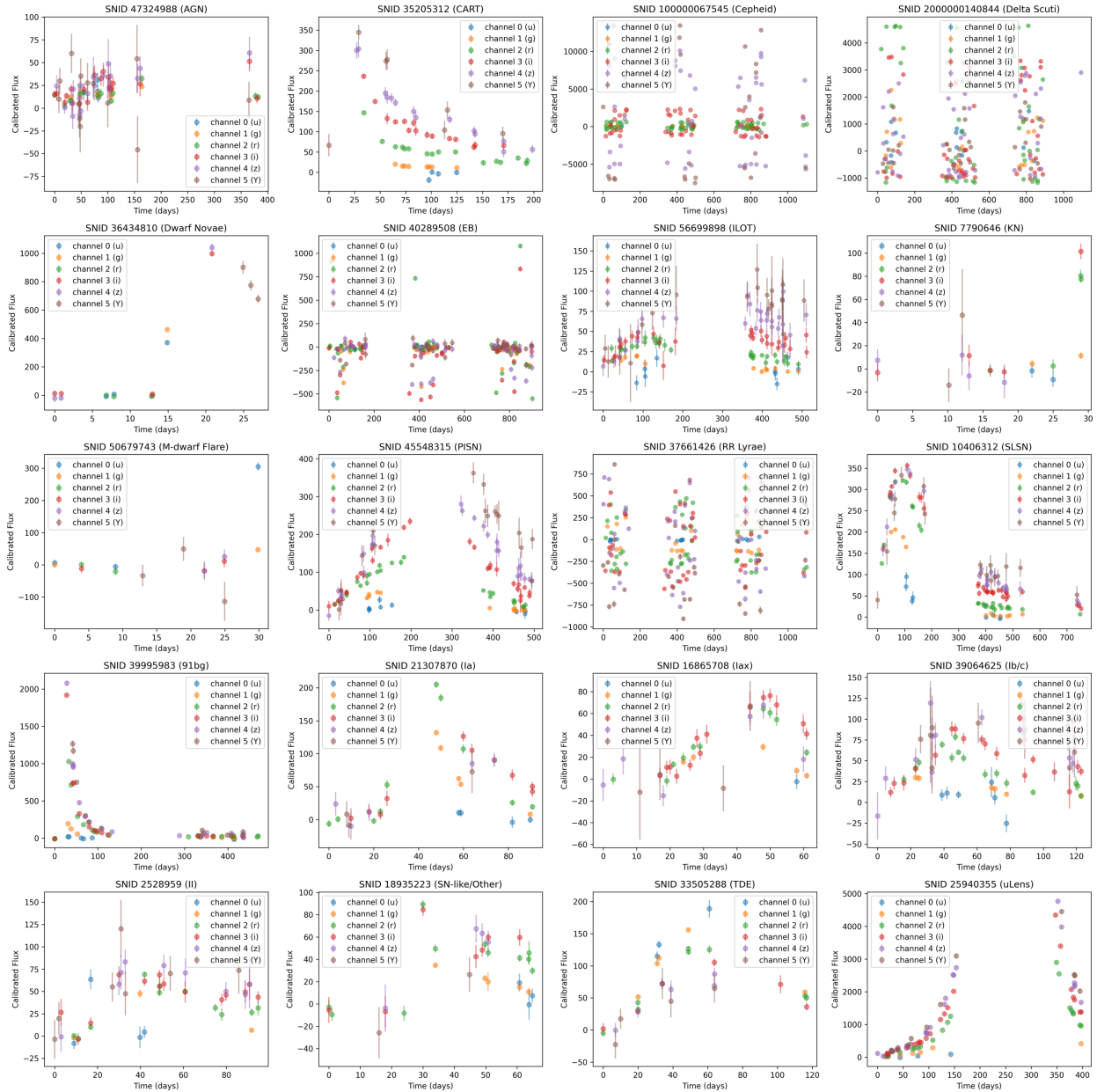


Figure 12. Class exemplars. For each class (using the ATAT 20-way classification scheme), we found the highest-scoring example in our validation set, using our LC-only model.

the left class classified as the right class, and examples from the right class classified as the left).

A.2 Reasonable scaling of ATAT’s single-core CPU results

ATAT reported their feature extraction CPU performance as a single-core number, but since processing batches of light curves is trivially parallelizable, we did a back-of-the-envelope calculation to scale it to a reasonable multi-core number.

Naive calculation and why it’s wrong: One might be tempted to multiply the performance by the number of cores in this chip (64), and then upscale by the nominal power consumption of the GPU (400/225), achieving a scalar of 114x. However, CPUs almost never linearly scale from single to multi-core performance, because (a) especially relevant for scientific tasks, the number of AVX and SSE units is often limited, resulting in reduced instruction-level parallelism; the chips also often downclock when many of these are used in parallel (b) L1, L2, and L3 caches are limited and

experience more pressure in multi-core workloads. The EPYC 7662 is also a newer chip than the A100.

Our calculation: The Ryzen 3995WX is a well-regarded CPU of the same generation as the A100, so we took its multi-core score according to cpubenchmark.net (83956), and divided it by the single-core score of the EPYC 7662 (2102), and then scaled by the nominal power consumption of the GPU (400/225), resulting in 57x scaling. We should also note that while we can monitor the GPU’s power consumption, we don’t know what the CPU’s would be like in this theoretical workload. CPUs report a “Typical TDP” which is sometimes less than their max/peak TDP in heavy workloads.

A.3 Empirical evaluation of model bias on simulator parameters

In Figure 15 and Figure 16, we examine the bias in two ELAS_{TiCC} simulator parameters, LOGMASS_{TRUE} and LOG_{SFR}, imparted by using quality cuts on our calibrated LC-only model, instead

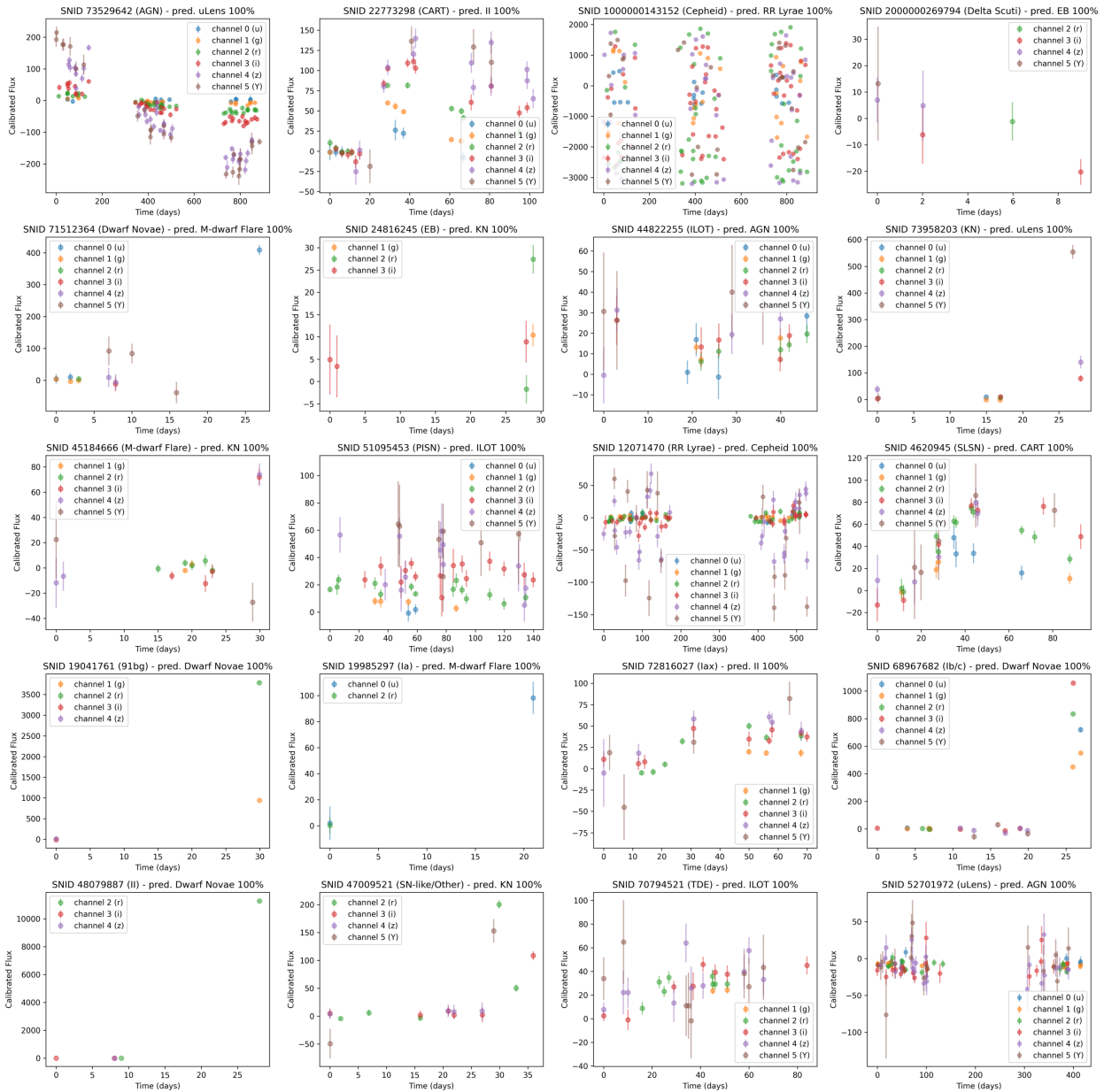


Figure 13. Examples strongly misclassified by our model (“class mis-exemplars”). For each class (using the ATAT 20-way classification scheme), we found the example in our validation set that scored the highest on *another* label, using our LC-only model. We can see that there is some trivial headroom in terms of the variable stars (RR-Lyrae, Delta-Scuti, Cepheid), and likely making our model incorporate period folding and/or Blazhko RR-Lyrae detection would improve those substantially. However, their F-1 scores are generally quite high. We did some preliminary experiments period folding all light curves and did not see a benefit, but believe this is because we need to do significance testing before period folding (and/or detect a distribution of likely periods).

of the true label. These parameters appeared to be shared by several supernova simulators, although they did not appear for many other classes. For this preliminary investigation, we simply removed the NaN values (corresponding to classes without these parameters). We would argue that the most adversarial values for the parameter on incorrectly-classified examples must not be extreme outliers (or else we could easily remove them), but still bias the distribution. Generally, these two parameters don’t suffer from much bias, despite having different distributions on classes and the fact that we used the LC-only model. The distribution shapes do not appear to change significantly, and the estimated mean from different samples is generally fairly controlled. The 91bg class showed sample means which were a small bit off from the true mean on both of these parameters. It would be valuable future work to see if different random initializations (or

bootstrappings) of this model would have the same consistent pattern. Please see Section 4.4 for general bias discussion.

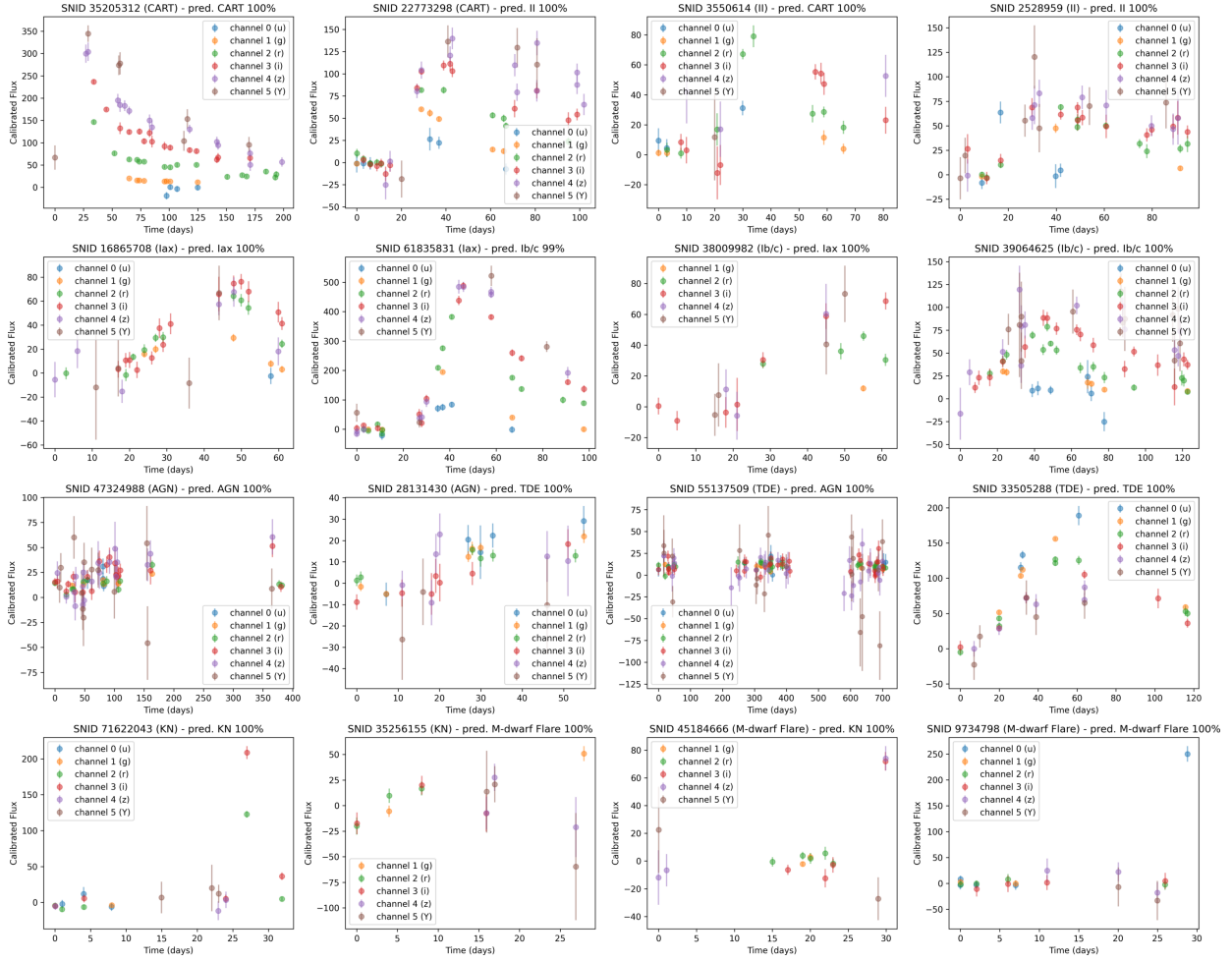


Figure 14. Pairwise confusions between classes. On the left column, we have the best class exemplar for one class i , and on the right column, we have the best class exemplar for another class j . Then, in the second (middle left) column, we have an exemplar from true class i that scored highest for class j , and in the third (middle right) column, we have an exemplar from true class j that scored the highest for class i .

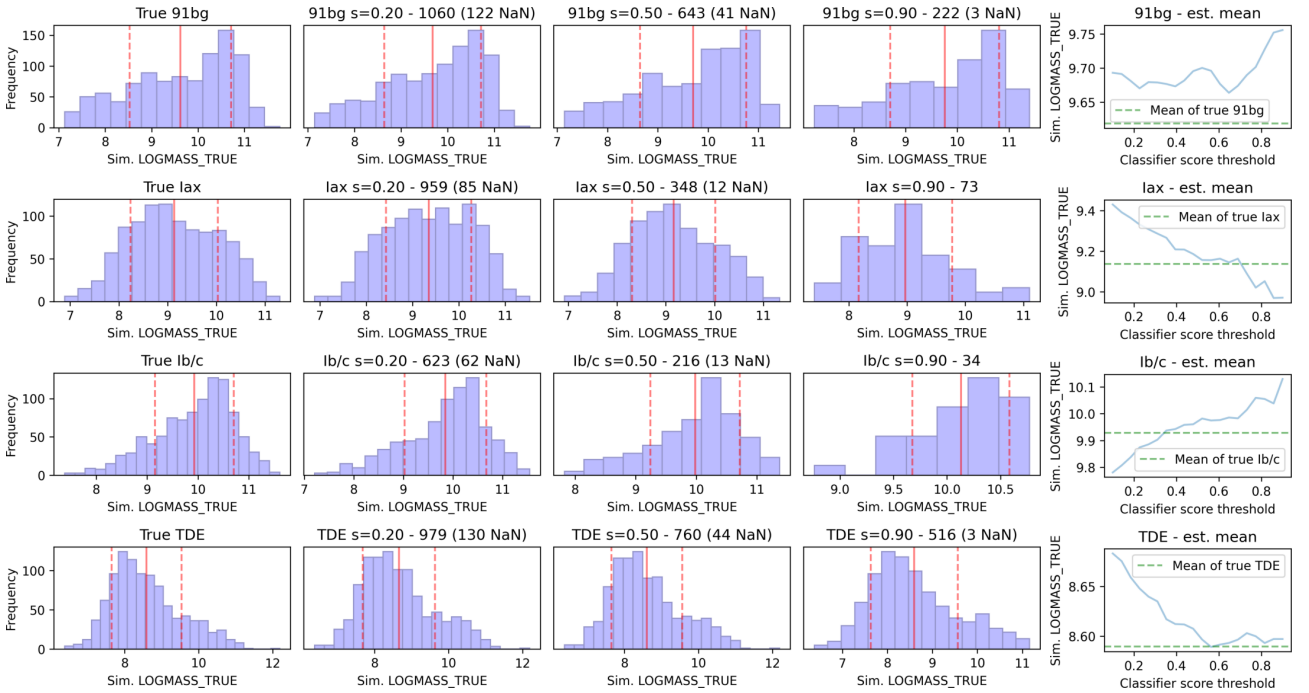


Figure 15. Empirical study of model selection bias effects on simulator parameter LOGMASS_TRUE. Left: True label distributions. Middle 3 columns: distributions when selecting by classifier probability score, using our calibrated LC-only model. Right: mean estimate vs. score threshold. The “s=” is the minimum score threshold, and after the “-” the total number of objects selected is shown. In parenthesis is the number of NaNs (from classes without LOGMASS_TRUE values), which we removed for this preliminary experiment.

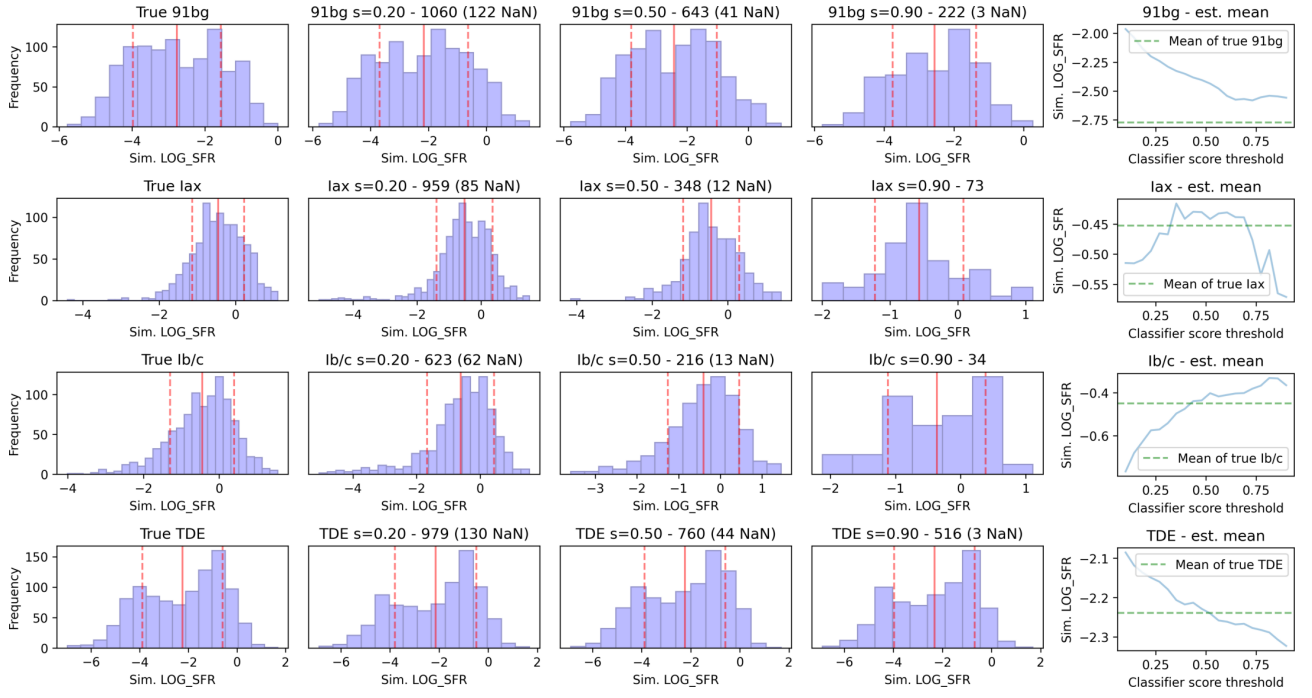


Figure 16. Empirical study of model selection bias effects on simulator parameter LOG_SFR . Please see Figure 15 for the format of this figure.