Variable Star Classification with Deep Learning

Sage Santomenna
Pomona College

Jalal Mufti
Vassar College

I. Introduction

Variable stars (variables) are stars that demonstrate an observable variation in brightness as a function of time. The physical mechanisms behind these variations is a subject of ongoing study, and varies across the different recognizable classes of variable stars. Studying variables serves many scientific purposes - for example, observation of variable stars can provide insight into the formation history of our galaxy [1]. Perhaps the most consequential outcome of the study of variable stars was Henrietta Swan Leavitt's discovery of the period-to-luminosity relationship present in Cepheid variables [2] [3]. This relation was used by Edwin Hubble to measure the expansion rate of the universe and continues to be an essential rung on the distance ladder that enables modern cosmology [4].
Essential to the study of variable stars is the ability to classify them from observational data. One method for doing so is to use features in spectral observations of variables as a fingerprint. This method, while effective, does not scale well and is intractable at large scale. Instead, a more efficient (but much more challenging) method for large-scale variable classification is to identify the stars by how their brightness changes with time. These timeseries observations of apparent luminosity (light curves) can be performed at scale by automated observatories, producing vast volumes of data. The challenge then becomes developing methods of variable star classification that don't rely on human identification. In this paper we develop deep learning models that attempt to perform this classification task.

II. Previous Work

One major source of inspiration and guidance for this project was a paper published in 2018 entitled "The ASAS-SN Catalog of Variable Stars I: The Serendipitous Survey", in which the authors aimed to classify variables stars using observations collected during supernova searches [5]. To classify these stars, the team utilized a machine learning approach. First, they calculated a set of features from each light curve and built a training set using labeled variables stars from existing catalogs, such as OGLE and AAVSO VSX. They then trained their own classifier on the data, a random forest classifier. This was used to classify newly identified stars, the results of which were then verified by experts. The final outcome of this research was the ASAS-SN Catalog of Variable Stars, a dataset of labeled variable stars, which we made use of in training and evaluating our model.

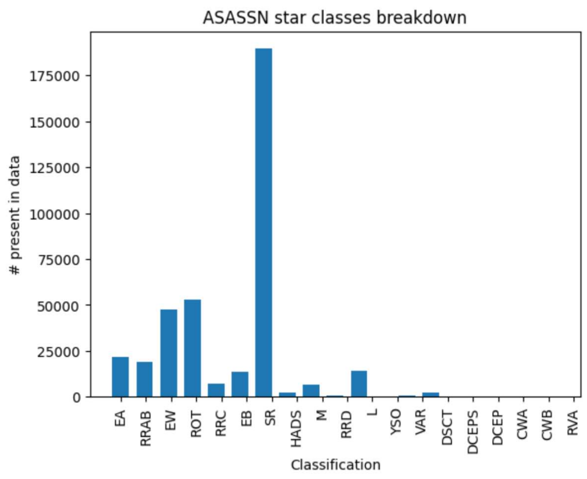


Fig. 1. The class distribution in the ASAS-SN dataset.

Additionally, the 2023 paper "Periodic variable star classification with deep learning: handling data imbalance in an ensemble augmentation way" was very influential [6]. In the work, the authors compared the performance of an RNN model, a CNN model, and a combination RNN and CNN model on the variable star classification task. In the process, they addressed the challenge of vast class imbalance by supplementing the training set with artificial lightcurves generated using gaussian processes. Their combined model achieved a macro F1 of 0.75.

III. Dataset

Our dataset was the ASAS-SN Catalog of variable star light curves. Reference [5] provides access to the data. As detailed in [5], the dataset contains 378,319 variable stars with over 180 million total observations. Each star belongs to 1 of 19 unique classifications; however, not all classes are evenly represented (fig. 1). The most populated class, with around 175,000 entries, has more than triple the amount of entries as the second most populated. On the other end, many classes contain fewer than 100 hundred entries, with the most extreme example being a class to which only 3 stars belong. Because of this, it was clear to us that we needed to handle the class imbalance in some way before the data could be passed to our model.

IV. Method

A. Data Preparation

To prepare the data for our model, we applied a number of processing steps to our light curves, including:

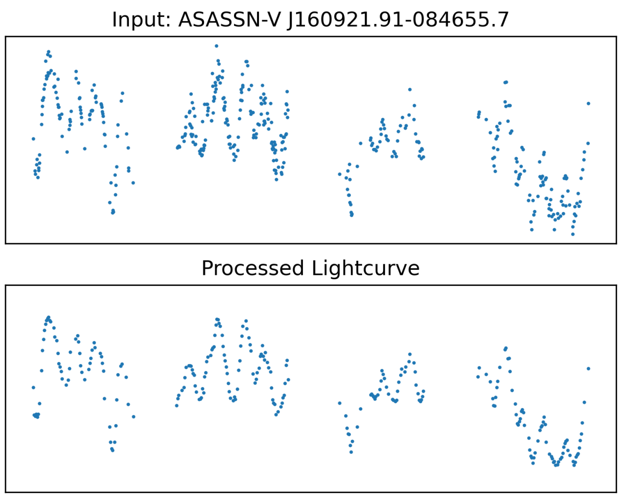


Fig. 2. An example of a light curve before and after processing.

1) Used sigma clipping, GHKSS [7], and SuperSmoother [8] to drop outlying points and smooth the noisy, over-sampled signal
2) Downsampled over-sampled light curves
3) Limited our training set to a class-balanced set of 44000 of the light curves belonging to classes with at least 5000 samples to remedy the extreme class imbalances present within the dataset and address computational constraints

An example of the effect of our data preparation process is shown in fig 2.

One challenge presented by the data arose from the following two facts:

1) The stars observed in the dataset are at a variety of distances from Earth
2) Two stars of equal *intrinsic* luminosity will appear to differ in brightness when observed from different distances away.

From this, a potential problem: the variation in amplitude and mean of the light curves is partially a function of distance. We're trying to determine star classification, which is not a (meaningful) function of distance, though distance does skew class frequency. If we do not correct this, the model could learn the wrong features from the data.
We can correct this using information about stellar distances from the Gaia spacecraft [9].

We performed these calculations, detailed in the appendix, on each light curve in the dataset.

Finally, we calculated a series of metrics called catch 22 [10] over each light curve, producing a representation of each light curve as a 1-dimensional vector of scalars. This representation could be interpreted by the MLP model we created (while the timeseries data could not).

B. Gaussian Processes

We also attempted to remedy the class imbalance through data augmentation using gaussian processes. In the appendix we include a brief primer on gaussian processes. We implemented the code to generate synthetic light curves from real ones (fig. 3), but elected not to use them to substitute our dataset until we developed a model that we were confident could handle a smaller, balanced dataset consisting solely of real light curves.

C. Initial MLP architecture

Pictured in figure 4 is our initial model architecture. It begins with an input layer followed by a sequence of dense layers, each followed by a LeakyReLU activation layer and a dropout layer for regularization. The architecture also includes residual connections, which help reduce vanishing gradient issues and enable better information flow through the network.

D. RNN

We experimented with a model that would use a bidirectional Gated Recurrent Unit (GRU) layer to perform the variable classification directly from the light curve timeseries (instead of from the metrics). Due to time and compute constraints, this solution space was not comprehensively explored and as a result we will keep our description of the model and its results brief.

E. Baseline: Random Forest

To gauge the effectiveness of using a deep learning model, we created a baseline model with which we could compare the results. Similar to the ASAS-SN paper, We opted for using a random forest classifier. We passed into it the same training, validation, and testing sets as our MLP model.

V. Evaluation

Our primary evaluation metric was the F1 score, which provides a balance between precision and recall scores. Since our classification task did not favor false positives over false negatives or vice versa, the F1 score was a suitable choice for assessing the model's performance. We used the macro F1 of each model over the test set to compare their efficacy.

VI. Results

A. Baseline: Random Forest

We used the default TensorFlow `RandomForestModel` as a baseline model for comparison. The model achieved a macro F1 of 0.645. Figure 5 gives the confusion matrix for our Random Forest model over the test set.

B. MLP

Our initial model's performance fell short of our expectations. While our random forest classifier was able to reach an F1 score of 0.645, the neural network only reached a score of 0.118, far below what we had hoped. After reevaluating, we realized that our model may have been over-engineered.

To address this, we simplified the architecture, reducing it to just three dense layers with two dropout layers for regularization. After training, this version of the model reached an F1 of 0.478, already a marked improvement from the initial design.

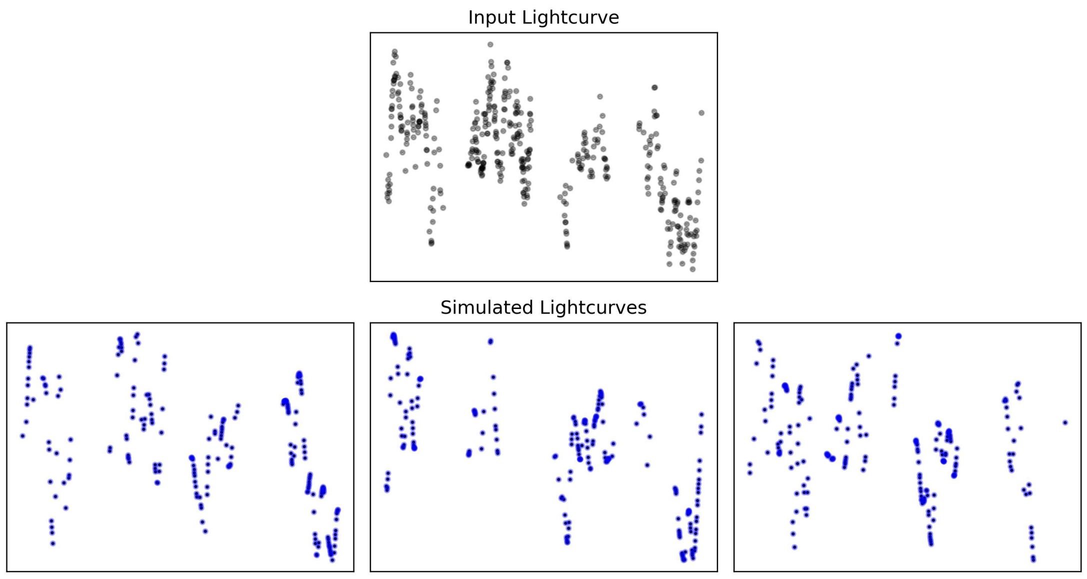


Fig. 3. An example of three synthetic light curves generated from a reference light curve.

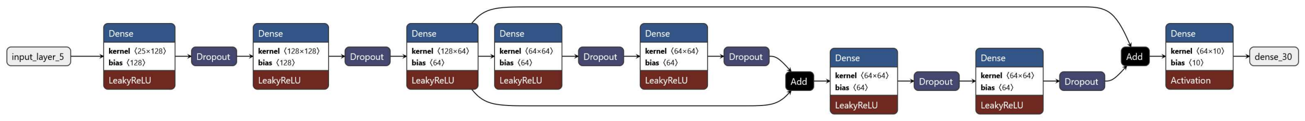
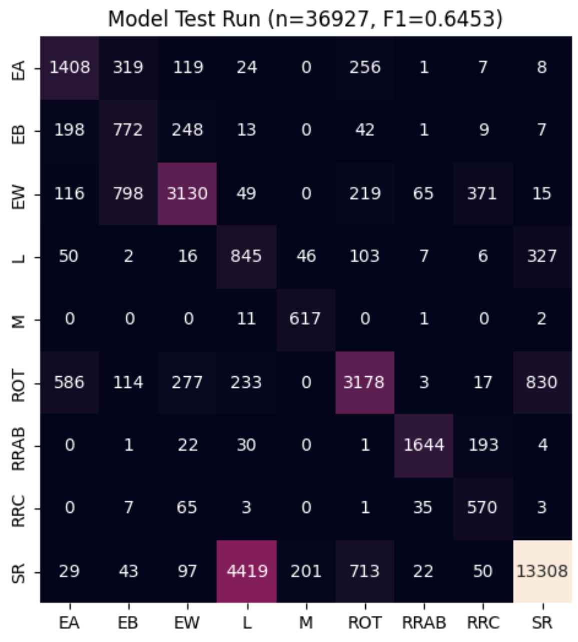


Fig. 4. The initial model design for our MLP model. This design proved to be insufficient and was iterated.


Model Test Run (n=36927, F1=0.6453)

	EA	EB	EW	L	M	ROT	RRAB	RRC	SR
EA	1408	319	119	24	0	256	1	7	8
EB	198	772	248	13	0	42	1	9	7
EW	116	798	3130	49	0	219	65	371	15
L	50	2	16	845	46	103	7	6	327
M	0	0	0	11	617	0	1	0	2
ROT	586	114	277	233	0	3178	3	17	830
RRAB	0	1	22	30	0	1	1644	193	4
RRC	0	7	65	3	0	1	35	570	3
SR	29	43	97	4419	201	713	22	50	13308

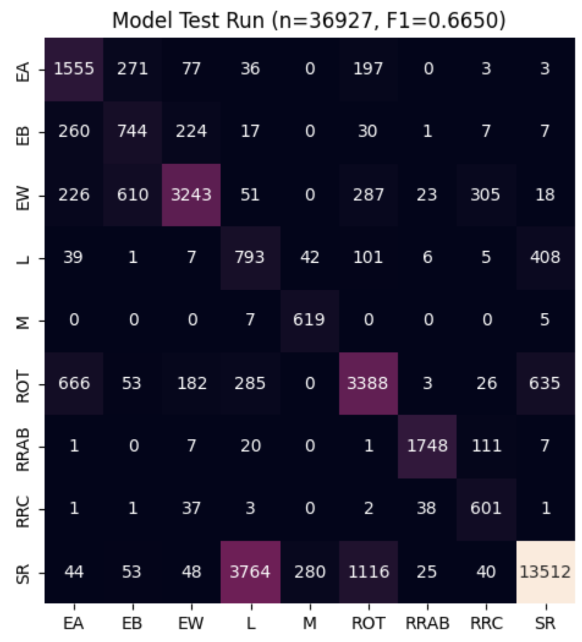
Fig. 5. The confusion matrix for our random forest model over the test set.

We then began a series of experiments to iteratively improve the model. Reducing the dropout rate from 0.5 to 0.2 offered a slight improvement (F1: 0.529), and lowering the batch size to 32 brought the score up further (F1: 0.565). This suggested that the lower dropout rate and smaller batch size were helping the model to not overfit and stabilizing the training.

Next, we experimented with normalization and layer sizing. Adding batch normalization after each dense layer yielded mixed results (F1: 0.535), but adjusting the hidden layer sizes had more impact. Increasing the size to 64 and 32 units resulted in an F1 score of 0.634, and further increasing to 128 and 64 brought us to 0.640, very close to our baseline performance. Finally, we reintroduced L2 regularization from the original architecture. This improved generalization and led to our best result yet, achieving an F1 score of 0.665, which narrowly outperformed the baseline random forest classifier. Figure 6 gives the confusion matrix for our MLP model over the test set. Figure 7 shows the architecture of the final model.

C. RNN

Our attempts at RNN models yielded little success. Our first approach was a simple RNN model with a masking layer, a 32-wide `SimpleRNN` layer, a dropout layer, and an output dense layer that padded sequences of lightcurve timestamps and magnitudes to all be the same length (the length of the longest training curve, 2419 timesteps). This proved disastrous, yielding a macro F1 of just 0.074. We tweaked our design to use a bidirectional Gated Recurrent



	EA	EB	EW	L	M	ROT	RRAB	RRC	SR
EA	1555	271	77	36	0	197	0	3	3
EB	260	744	224	17	0	30	1	7	7
EW	226	610	3243	51	0	287	23	305	18
L	39	1	7	793	42	101	6	5	408
M	0	0	0	7	619	0	0	0	5
ROT	666	53	182	285	0	3388	3	26	635
RRAB	1	0	7	20	0	1	1748	111	7
RRC	1	1	37	3	0	2	38	601	1
SR	44	53	48	3764	280	1116	25	40	13512

Fig. 6. The confusion matrix for our MLP model over the test set.

Unit (GRU) layer instead of a simple RNN, and to accept a small randomly selected continuous window of 15 points from each light curve instead of the whole thing. This yielded only a modicum of improvement with a macro F1 of 0.083. During our experimentation we frequently encountered compute limits that slowed our progress significantly.

VII. Discussion

A. MLP

We were pleased that we were able to develop a model that surpassed the baseline classifier's macro F1 (0.665 vs 0.645). Both the MLP and the random forest classifier showed similar behavior of incorrectly classifying some 'semi-regular' (SR) stars as 'irregular' (L). These classes are the most loosely-defined of the ASAS-SN classes, so it is unsurprising that it is difficult to draw a clear distinction between them. It could also be the case that our catch-22 metrics are not capturing information that would allow us to better discriminate between the two.

B. RNN

The performance of the experimental RNN model was disappointing but not very surprising. The first sequence length (2419 timesteps) was much more than what the RNNs could reasonably be expected to process, and the GRU window length (15 timesteps) was likely too small to capture all of the necessary information. In the future we will look into resampling the data to the same cadence before determining the appropriate window length by direct inspection of a sample of the light curves. This has the potential to improve the performance of the model significantly.

VIII. Summary

We detail our efforts to build deep learning models that can classify variable stars from observations of how their brightness changes with time. We primarily focus on designing an MLP model and also make small forays into an RNN-based architecture. We compare both to a baseline random forest model, achieving moderate success with the MLP, but lack the time and compute for the RNN model to come to fruition.

References

[1] V. D'Orazi, et al. "The GALAH survey: tracing the Milky Way's formation and evolution through RR Lyrae stars," *Monthly Notices of the Royal Astronomical Society*, vol. 531, no. 1, pp. 137-162, 2024, doi: 10.1093/mnras/stae1149.

[2] H. S. Leavitt, "1777 variables in the Magellanic Clouds," *Annals of Harvard College Observatory*, vol. 60, pp. 87-108.3, 1907.

[3] H. S. Leavitt and E. C. Pickering, "Periods of 25 Variable Stars in the Small Magellanic Cloud.," *Harvard College Observatory Circular*, vol. 173, no. pp. 1-3, 1912.

[4] E. Hubble, "A Relation between Distance and Radial Velocity among Extra-Galactic Nebulae," *Proceedings of the National Academy of Science*, vol. 15, no. 3, pp. 168-173, 1929, doi: 10.1073/pnas.15.3.168.

[5] T. Jayasinghe, et al. "The ASAS-SN catalogue of variable stars I: The Serendipitous Survey," *Monthly Notices of the Royal Astronomical Society*, vol. 477, no. 3, pp. 3145-3163, 2018, doi: 10.1093/mnras/sty838.

[6] Z. Kang, et al. "Periodic Variable Star Classification with Deep Learning: Handling Data Imbalance in an Ensemble Augmentation Way," *Publications of the Astronomical Society of the Pacific*, vol. 135, no. 1051, pp. 094501, 2023, doi: 10.1088/1538-3873/acf15e.

[7] R. Hegger, H. Kantz, and T. Schreiber, "Practical implementation of nonlinear time series methods: The TISEAN package," *Chaos*, vol. 9, no. 2, pp. 413-435, 1999, doi: 10.1063/1.166424.

[8] J. Vanderplas, "supersmoother: Efficient Python Implementation of Friedman's SuperSmoother". *Zenodo*, 2015, doi: 10.5281/zenodo.14475.

[9] Gaia Collaboration, et al. "Gaia Data Release 3. Summary of the content and survey properties," *Astronomy and Astrophysics*, vol. 674, no. pp. A1, 2023, doi: 10.1051/0004-6361/202243940.

[10] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "catch22: CAnonical Time-series CHaracteristics," *arXiv e-prints*, 2019, doi: 10.48550/arXiv.1901.10200.

[11] M. Guillame-Bert, S. Bruch, R. Stotz, and J. Pfeifer, "Yggdrasil Decision Forests: A Fast and Extensible Decision Forests Library," *arXiv e-prints*, vol. no. pp. arXiv:2212.02934, 2022, doi: 10.48550/arXiv.2212.02934.

Appendix A
Magnitude Normalization

Given the distance to a source, we can transform each recorded apparent magnitude m (logarithmic measure of apparent spectral flux density) to distance-invariant absolute magnitude M:

$$m = -2.5 \log_{10} \left(\frac{F_{obs}}{F_{\text{ref}}} \right)$$

where F_{obs} is the observed spectral flux density of the star and F_{ref} is the spectral flux density of the photometric system's reference zeropoint.
The absolute magnitude is

$$M = m - 5 \log_{10} \left(\frac{d}{10\text{pc}} \right)$$

For an object of apparent magnitude m at distance d from the observer.
We can find the uncertainty in the absolute magnitude is by

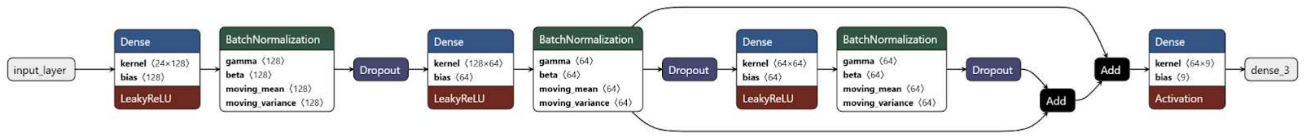


Fig. 7. The final model design for our MLP model.

combining some uncertainty Δd in the distance measurement with photometric (magnitude) uncertainty Δm:

$$\begin{aligned}\frac{\Delta M}{M} &= \sqrt{\left(\frac{\Delta d}{d}\frac{\partial M}{\partial d}\right)^2 + \left(\frac{\Delta m}{m}\frac{\partial M}{\partial m}\right)^2} \\ &= \sqrt{\left(\frac{\Delta d}{d}\frac{-5}{\ln(10)d}\right)^2 + \left(\frac{\Delta m}{m}\right)^2}\end{aligned}$$

Appendix B
Gaussian Processes

Our goal is to, given observations $\mathbf{X}$, a corresponding $\vec{y}$, and an unobserved $\mathbf{X}^*$, predict the unobserved output $\vec{y^*}$.
We assume that our data are made up of noisy observations of some function f that provides a distribution for y at each input x:

$$y_i \sim f(\mathbf{x}_i) + \epsilon_i$$

We'll make a prediction using GP's **core assumption** that $\vec{y}$ and $\vec{f^*}$ are jointly distributed as an (N+M) dimensional multivariate normal:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{K}}_{\mathbf{X},\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{x}^*} \\ \mathbf{K}_{\mathbf{X}^*,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^*,\mathbf{X}^*} \end{bmatrix}\right)$$

Where $K_{A,B}$ is a an N-by-N distance matrix between A and B, as computed by an input kernel $K(\vec{x_i}, \vec{x})$
We account for our data by considering a prior of an infinite number of different possible fs that could represent the data and then conditioning.
Luckily, the multivariate norm has a well-defined posterior over $\vec{f^*}$:

$$\mathbf{f}^* \mid \mathbf{X}^*, \mathcal{D} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{f}^*}, \boldsymbol{\Sigma}_{\mathbf{f}^*}\right)$$

Where $\boldsymbol{\mu}_{\mathbf{f}^*}$ is a mean (derived from the distance matrix and the data) and $\boldsymbol{\Sigma}_{\mathbf{f}^*}$ is the covariance, also derived from the distance matrix.
The posterior predicts a distribution $p(y'|\vec{x_i^*})$ at each desired prediction point x^* with its mean and cov determined by $\boldsymbol{\mu}_{\mathbf{f}^*}$ and K. We can simulate our light curves by drawing from this distribution for each test x, then adding noise.