Draft version April 8, 2026
Typeset using LaTeX **twocolumn** style in AASTeX631

ASTRAFier: A Novel and Scalable Transformer-based Stellar Variability Classifier

Paul F. X. Gregory [1], Jeroen Audenaert [1], Mykyta Kliapets [1,2], Daniel Muthukrishna [1,3], Andrew Tkachenko [2], Marek Skarka [4], Marc Hon [5,1] and George R. Ricker [1]

[1] *MIT Kavli Institute for Astrophysics & Space Research, Massachusetts Institute of Technology, Cambridge, MA, USA*
[2] *Institute of Astronomy, KU Leuven, Celestijnenlaan 200D, bus 2401, 3001 Leuven, Belgium*
[3] *AstroAI, Center for Astrophysics | Harvard & Smithsonian, 60 Garden Street, Cambridge, 02138, MA, USA*
[4] *Astronomical Institute of Czech Academy of Sciences, Fričova 298, 251 65 Ondřejov, Czech Republic*
[5] *Department of Physics, National University of Singapore, 21 Lower Kent Ridge Road, Singapore, 119077*

ABSTRACT

Photometric missions such as *Kepler* and TESS have generated millions of light curves covering almost the entire sky, offering unprecedented opportunities to study stellar variability and advance our understanding of the Universe. In this data-rich environment, machine learning has emerged as a powerful tool to efficiently and accurately process and classify light curves according to their type of stellar variability. In this work, we introduce ASTRAFier: a novel Transformer-based model for variability classification that integrates Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Networks (CNNs). The model operates directly on time series without requiring feature engineering, creating an easy-to-maintain and efficient end-to-end classification framework. We train and validate our model using both *Kepler* and TESS light curves and, respectively, achieve a classification accuracy of 94.26% on *Kepler* and 88.22% on TESS. We demonstrate scalability by deploying our model on $\sim$ 2.8 million TESS light curves from sectors 14, 15, and 26 (Kepler Field-of-View) delivered by MIT's Quick-look Pipeline (QLP) and release the resulting stellar variability catalog.

Keywords: methods: data analysis, methods: statistical, techniques: photometric, stars: variables

1. INTRODUCTION

The temporal variability of stars can reveal their interior workings, evolutionary pathways and the presence of companion objects, while large-scale analyses can provide insights for population studies (e.g., Aerts et al. 2010; Aerts 2021; Kurtz 2022). Stellar variability and asteroseismology have been revolutionized with the advent of space missions such as *Kepler*/K2 (Borucki et al. 2010; Koch et al. 2010; Howell et al. 2014) and the Transiting Exoplanet Satellite Survey (TESS, Ricker et al. 2015), delivering millions of uninterrupted high-quality light curves (e.g., Huber 2025).

By now, TESS has observed nearly the entire sky in sectors of 27.4 days. The Full-Frame Images (FFIs) have 30-min, 10-min and 200-sec cadence for the primary (PM), first extended (EM1) and second extended (EM2) mission, respectively. The total observing baselines ranging from a few months to multiple years in the Continuous Viewing Zone, where the recently started third extended mission (EM3) also includes a number of 54 day sectors. The upcoming PLAnetary Transits and Oscillations of stars (PLATO, Rauer et al. 2024) mission will be launched in 2027 and continuously observe the same patch of sky for at least two years of time (Nascimbeni et al. 2025; Jannsen et al. 2025).

The sheer scale of the data necessitates efficient and effective automated analysis methods (e.g., Audenaert 2025). The classification of stars according to their variability type is essential for building large samples of stars for detailed astrophysical analyses, identifying promising targets for follow-up observations, and informing future space missions (e.g., Eschen et al. 2024, who studied the PLATO field-of-view using TESS).

Variability catalogs for TESS have been constructed using statistical and visual methods for subsets of TESS observations (e.g., Skarka et al. 2022; Fetherolf et al.

jeroena@mit.edu, paulg9@mit.edu

2023; Skarka & Henzl 2024; Kemp et al. 2025). Additionally, dedicated classification methodologies relying on machine learning and statistical techniques have been created for identifying solar-like oscillators (e.g., Hon et al. 2018b,a, 2019; Nielsen et al. 2022; Hatt et al. 2023), eclipsing binaries (e.g., IJspeert et al. 2021, 2024b,a), short-period variables (e.g., Olmschenk et al. 2024), transients (e.g., Roxburgh et al. 2025) and pulsators (e.g., using both TESS and Gaia, Hey & Aerts 2024).

Machine learning has proven to be the most effective technique for performing large-scale automated classifications across a wide range of variability classes (e.g., Jamal & Bloom 2020; Audenaert et al. 2021; Huijse et al. 2025; Audenaert 2025). Traditionally, supervised classification methodologies mostly relied on feature engineering techniques to characterize the properties of light curves, for example, with features derived from statistical moments, Lomb-Scargle periodogram (Lomb 1976; Scargle 1982) and entropy (Shannon 1948), such as those in Choi et al. (2025). The features are then fed as input to, for example, random forests (Breiman 2001), gradient boosting machines (Friedman 2001), Gaussian mixture models or Convolutional Neural Networks (CNN) (e.g., Debosscher et al. 2007; Sarro et al. 2009; Blomme et al. 2011; Richards et al. 2011; Kim & Bailer-Jones 2016; Armstrong et al. 2016; Hon et al. 2018b; Barbara et al. 2022; Cui et al. 2024). In addition to supervised approaches, unsupervised settings have been increasingly explored to handle the growing volume of data; for instance, Audenaert & Tkachenko (2022) used entropy-based features in an unsupervised setting, while Ranaivomanana et al. (2025) and Huijse et al. (2025) utilized dimensionality reduction and deep representation learning via autoencoders, respectively, to discover and classify variable sources without prior labeling. Audenaert et al. (2021) combined multiple distinct models, each relying on different feature sets, into an ensemble classification model to achieve a higher performance.

Automated representation learning models (see Audenaert 2025, for an overview) have been used to learn the characteristic features of light curves for variability classification. Naul et al. (2018); Becker et al. (2025) used Recurrent Neural Networks (RNNs) to classify sparse light curves, while Muthukrishna et al. (2019) used RNNs with gated recurrent units (GRUs) to classify transients.

Since their introduction, Transformers (Vaswani et al. 2017) have become a cornerstone in Generative Artificial Intelligence (AI) and natural language processing, powering models such as ChatGPT (Radford et al. 2018) and BERT (Devlin et al. 2018). Their success in NLP has spurred interest in applying Transformer architectures to time series data, where their capacity to learn dependencies and correlations between sequence elements offers promising advantages (Wen et al. 2022). Pan et al. (2024) used a transformer along with a CNN to predict log g values from light curves. The use of the transformer was found to increase performance of the model over just a CNN, especially in capturing long term dependencies, with other examples being Donoso-Oliva et al. (2023); Rizhko & Bloom (2025); Moreno-Cartagena et al. (2025); Donoso-Oliva et al. (2026).

In this work, we present a novel machine learning framework to classify stars according to their variability classes. Our model, named ASTRAFier (Astronomical Sequence TRansformer-based vAriability classifier), utilizes LSTM, Transformers, and CNNs to process the light curve, offering a powerful architecture for classification. This architecture is designed to directly process raw light curve data, eliminating the need for feature engineering while effectively capturing the complex temporal patterns inherent in stellar variability. We build on the earlier classification work by the TESS Asteroseismic Science Consortium (Audenaert et al. 2021) and leverage their training set of *Kepler* light curves and its cross-match with TESS.

We give a theoretical overview of the different machine learning components in Sect. 2, discuss our model in Sect. 3, training set in Sect. 4 and training procedure in Sect. 5. We analyze the results on our labeled training set in Sect. 6 and deploy our model to all light curves in TESS sectors 14, 15, and 26 in Sect. 7 to obtain a catalog of variable star candidates. These sectors are of particular interest as they provide spatial overlap with the Kepler field of view.

2. BACKGROUND

This section introduces the fundamental machine learning components behind our model in a light curve processing context: Transformers, CNNs, and LSTMs.

2.1. *Transformers*

The main mechanism behind the Transformer is multi-head self-attention (MHSA, Vaswani et al. 2017). Self-attention works by transforming each token (a unit of data, for language models typically a word or part of a word, and in the case of light curves a time step) into three learned representations via linear projections; the query, key, and value. In short, the query seeks relevant context from other tokens. The key indicates how suitable a token is in responding to queries from other tokens. The value contains the content of the token that is weighted and aggregated based on how well the key

matches the query, determining which parts of the original sequence influence the output. The attention matrix is computed as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{\mathrm{d_k}}})\mathbf{V}, \quad (1)$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices, respectively, and $\mathrm{d_k}$ is the dimension of the key matrix.

This computes the relevance of each position in the sequence to every other position, telling the model where it should pay more attention (i.e., the "attention" mechanism). For multi-head attention, multiple self-attention mechanisms are employed in parallel with independently learned key, query, and value matrices, and these outputs are then concatenated, enabling the model to learn more complex relationships as different heads can focus on different parts of the input. The parallelism of the multiple heads also allows for more efficient computations.

To incorporate the sequential order of a time series into the model, as Transformers are inherently permutation-invariant, positional encodings are added to the input embeddings. The original Transformer architecture (Vaswani et al. 2017) introduced sinusoidal positional encodings, which alternate sine and cosine functions of varying frequencies across adjacent dimensions to encode absolute position. This approach has two key advantages: it allows the model to extrapolate to sequence lengths longer than those seen during training, and the sinusoidal structure enables the model to learn relative positions through linear projections.

However, the standard positional encoding assumes uniformly spaced inputs, which is not guaranteed for astronomical time series. Light curves often contain gaps due to spacecraft operations, data quality cuts, or observing constraints. To address this, we derive our positional encoding directly from the time vector of the input light curve rather than using integer position indices (Zuo et al. 2020), ensuring that the encoding reflects the true temporal spacing between observations. We additionally scale the input to the sine and cosine functions in the positional encoding by d_{emb}/T, following Foumani et al. (2023), which prevents the positional encodings from becoming indistinguishable when the embedding dimension is small relative to the sequence length. We apply these two modifications to the original encoding of Vaswani et al. (2017), yielding the following:

$$\mathbf{PE}_{(\mathrm{pos},2i)} = \sin\left(\frac{\mathrm{pos}}{10000^{\,2i/d_{\mathrm{emb}}}} \cdot \frac{d_{\mathrm{emb}}}{T}\right) \quad (2)$$

$$\mathbf{PE}_{(\mathrm{pos},2i+1)} = \cos\left(\frac{\mathrm{pos}}{10000^{\,2i/d_{\mathrm{emb}}}} \cdot \frac{d_{\mathrm{emb}}}{T}\right) \quad (3)$$

where pos is the observation timestamp, $i \in \{0, \ldots, d_{\mathrm{emb}}/2 - 1\}$ along the embedding dimension, T is the number of time steps, and d_{emb} is the embedding dimension. $\mathbf{PE}$ has shape (T, d_{emb}) and is added element-wise to the Transformer input, ensuring that temporal order information is preserved.

A Transformer encoder block consists of a positional encoding followed sequentially by MHSA and a feed-forward (a network of non-linear transformations flowing in one direction) module. Residual connections are applied around both the self-attention and feed-forward modules, as illustrated in Fig. 1.

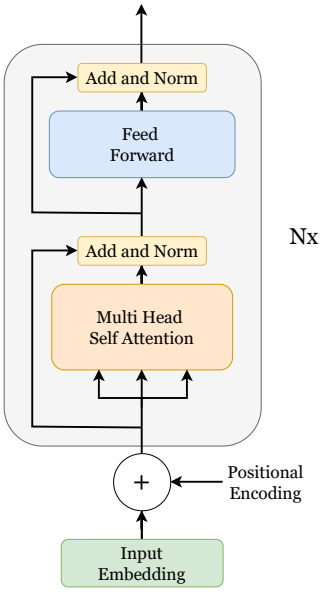

Figure 1. A Transformer encoder layer. Figure reproduced from Vaswani et al. (2017).

2.2. *Long Short-Term Memory (LSTM)*

The foundation for LSTMs (Hochreiter & Schmidhuber 1997) was laid by Recurrent Neural Networks (RNNs). Unlike feedforward neural networks, RNNs are designed to process sequences of data by maintaining a hidden state that evolves over time. At each time step t, the network updates its hidden state h_t based on the current input and the previous hidden state h_{t-1}. This recurrent connection allows the network to retain information from earlier time steps. A significant limitation of traditional RNNs is the vanishing gradient problem, where the influence of earlier inputs diminishes as gradients are backpropagated through many time steps, hindering the ability of RNNs to process long sequences.

LSTMs address this issue through the use of a cell state that can retain important information over long durations, ensuring that long-term dependencies are not forgotten as the sequence progresses. In short, the cell state handles long-term memory, while the hidden state handles short-term memory. The LSTM uses three gates to control what information is remembered: the forget gate, the input gate, and the output gate. The forget gate determines what parts of the previous cell state can be discarded, the input gate decides how much of the new input should be added to the cell state, and the output gate regulates the influence of the cell state on the current hidden state. This gating mechanism enables LSTMs to preserve important information over extended sequences. An LSTM block for a single time step can be seen in Fig. 2.

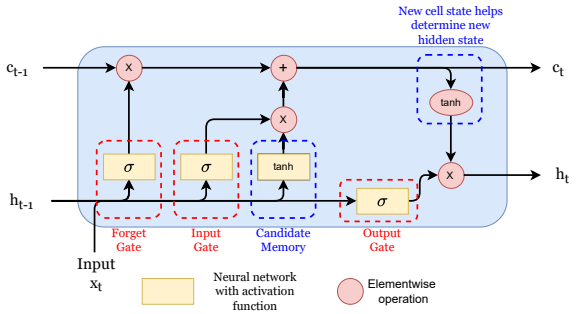

Figure 2. An LSTM block at time step t. x_t is index t of the input sequence, c_t is the cell state at time t, and h_t is the hidden state at time t. An LSTM module consists of many such blocks, typically one for each time step in the input sequence. The LSTM outputs its hidden states $[h_1, h_2, ..., h_T]$.

In our model, we make use of a bidirectional LSTM (BiLSTM, Schuster & Paliwal 1997). This expands on the LSTM by processing the input sequence in both the forward and backward directions. Essentially, one LSTM reads the sequence from start to end, another LSTM reads it from end to start, and these outputs are concatenated, allowing the model to leverage information from both past and future contexts. This dual perspective is particularly advantageous for time series classification, as it enables the capture of dependencies in both temporal directions.

2.3. *Convolutional Neural Networks (CNNs)*

CNNs (LeCun et al. 1998) are a feed-forward architecture proficient at handling grid-like data structures. Originally popularized in computer vision for tasks such as handwritten digit recognition (LeCun et al. 1989, 1998) and large-scale image classification (Krizhevsky et al. 2012), CNNs have also demonstrated significant utility in processing time-series data by treating sequences as one-dimensional grids to capture temporal patterns (Wang et al. 2016). CNNs consist of convolutional layers that employ learnable filters, called kernels, to capture spatial hierarchies and extract local features from the input. The kernel slides across the input, transforming it with the values it has learned to produce the output. To handle the edges of the data, the input is padded with values, often zeros, that allow the center of the kernel to reach the edges. A standard CNN layer consists of a convolution, batch normalization, and an activation function.

In time-series data, 1-D convolutions can be useful in detecting temporal patterns. An example of a 1-D kernel convolving an input sequence to produce a 1-channel output can be seen in Fig. 3.

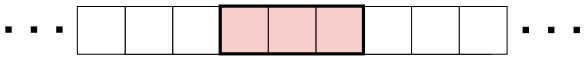

Figure 3. A 1-D kernel of size 3. This kernel slides along the input sequence *stride* steps at a time, producing a new sequence through a multiplication of its learned weights and the input sequence.

While this example shows a CNN limited to handling 1-channel inputs and 1-channel outputs, we can generalize to handle multi-channel inputs and outputs as well. To handle a multi-channel input, we use a multi-channel kernel, filtering each channel in the input with the corresponding channel in the kernel, summing the outputs from each channel to get our 1-channel output. To handle a multi-channel output, we use a set of filters, called a filter bank. To get C_{out} output channels, we use a filter bank of C_{out} filters, one for each output channel. We combine these two ideas to be able to handle multi-channel inputs and outputs.

As an example, take an input with C_{in} channels $\mathbf{x}_{in} \in \mathbb{R}^{C_{in} \times T}$. To get an output with C_{out} channels, we compute

$$\mathbf{x}_{out}[c_2, :] = \sum_{c_1=1}^{C_{in}} \mathbf{w}[c_1, c_2, :] * \mathbf{x}_{in}[c_1, :] \tag{4}$$

Where $\mathbf{x}_{out} \in \mathbb{R}^{C_{out} \times T}$ is our output, $c_2 \in \{0, ..., C_{out} - 1\}$ indexes the output channel, and $\mathbf{w} \in \mathbb{R}^{C_{in} \times C_{out} \times K}$ is our filter bank of size K kernels. Fig. 4 shows a visualization of this.

In our CNN layers, we make use of the Gated Linear Unit activation function according to

$$GLU(a, b) = a \otimes \sigma(b) \tag{5}$$

where a is the first half of the input and b is the second half, and σ is the sigmoid function. This activation

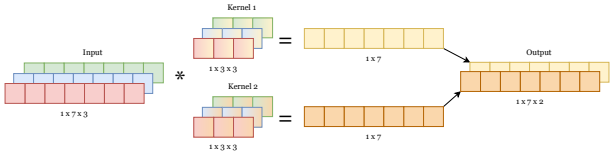

Figure 4. A visualization of a 1-D convolution with 3 input channels and 2 output channels.

function has been found to improve performance when modeling sequential data (Dauphin et al. 2016).

3. MODEL ARCHITECTURE

Recent research has shown the advantages of integrating attention mechanisms, CNNs, and LSTMs due to their complementary capabilities in handling sequential data (e.g., Shen et al. 2024; Zhang et al. 2023; Ranjbar & Rahimzadeh 2024). Transformers excel at capturing global context through self-attention, while CNNs specialize in detecting localized features via convolutional filters, and LSTMs manage sequential memory and long-term dependencies.

Our model, ASTRAFier, is shown in Fig. 5 and is a novel sequential hybrid architecture that integrates BiLSTM, Transformer, and CNN modules with residual connections. The residual connections add a module's input to its output and normalize the sum and are shown by the "Add and Norm" blocks. Our design enables each component to collaboratively process the information contained in a light curve, while the residual connections ensure that the characteristic information extracted by each module is preserved and effectively propagated throughout the network.

The light curve input is embedded and processed through three sequential ASTRAFier blocks (gray box in Fig. 5). The outputs of these blocks are then averaged across the time dimension and passed through a Multi-Layer Perceptron (MLP) with a softmax activation function for the final output probabilities for each class.

3.1. *Handling Variable-Length Sequences*

Astronomical light curves vary in length due to differences in observing strategy, instrument design and data quality flags. When processing batches of variable-length sequences, shorter sequences are padded to match the longest sequence in the batch. To prevent these padded positions from influencing the model's learned representations, we propagate binary padding masks throughout key stages of the network. These masks indicate valid observations ($m_t = 1$) versus padding ($m_t = 0$). Attention scores for padded positions are set to $-\infty$ before softmax, LSTM outputs at padded

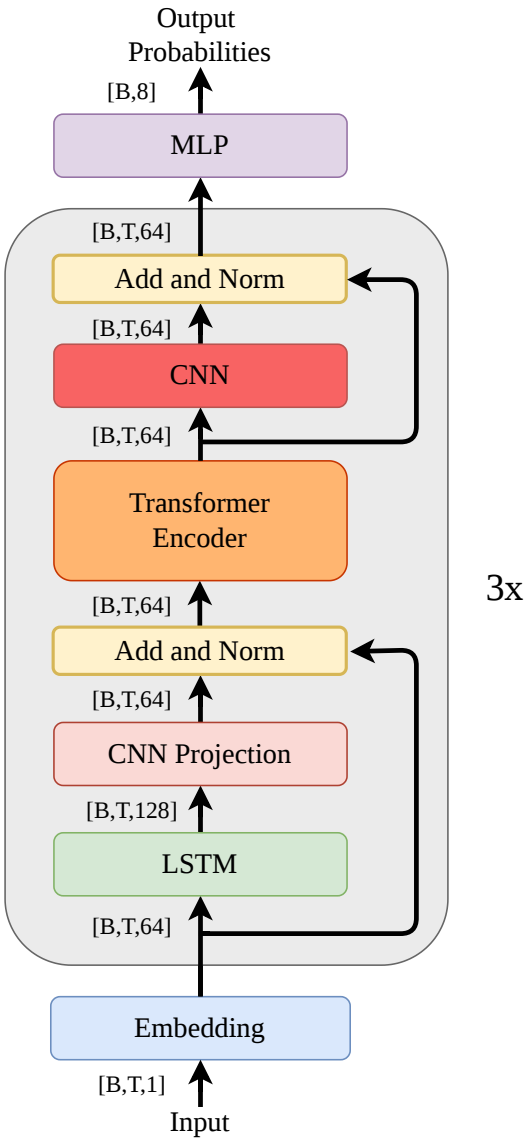

Figure 5. Architecture of the ASTRAFier model. The gray box highlights a single block, which is stacked three times. Each block contains a BiLSTM (with a CNN Projection), a Transformer encoder, and a CNN module. *Add and Norm* refers to a residual (skip) connection followed by layer normalization, which facilitates gradient flow and stabilizes training. Positional information is injected into the initial embeddings using the time-dependent sinusoidal encodings described in Eqs. 2-3. Tensor shapes are annotated between modules in the form $[B, T, C]$, where B is the batch size, T the number of time steps, and C the feature dimension.

positions are zeroed, and final sequence representations are computed via masked averaging rather than standard global pooling. The convolutional layers do not explicitly apply the masks (in line with, e.g., TorchAudio 2025). Our normalization choice in these layers ensures the normalization for each valid frame is computed independently of any of the padded positions (see Sects. 3.3 and 3.5 for details). The residual effects are only confined to local boundary overlaps from the convolutional kernels, in contrast to a global effect that would be caused by unmasked attention. The padded positions are excluded from the output entirely because the final representation uses masked averaging (Eq. 6).

3.2. *Embedding*

The light curves are first embedded using a fully connected layer that maps a single input feature (a scalar representing the flux at a particular time step) to 64 output features (our chosen embedding dimension). Essentially, this takes the input flux vector ($\mathbb{R}^{\mathrm{TimeSteps}}$), and for every time step, projects it into a 64-dimensional vector ($\mathbb{R}^{64}$) via a linear transformation, yielding a new embedded light curve ($\mathbb{R}^{TimeSteps\times 64}$). This higher dimensional representation allows the model to learn a richer representation of the data. Fig. 6 illustrates the embedding process.

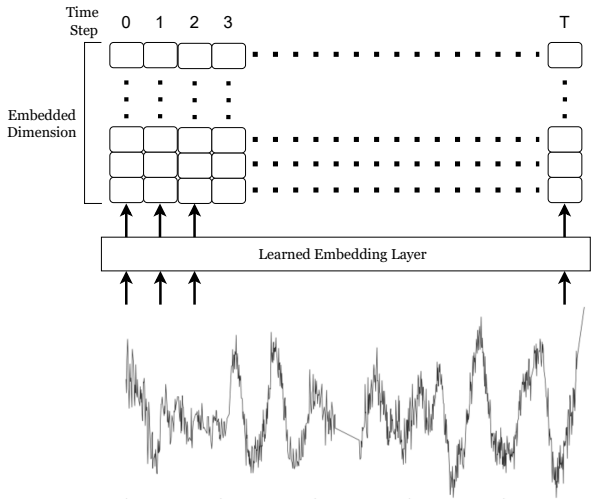

Figure 6. A preprocessed light curve is embedded into a higher dimension.

3.3. *BiLSTM Module*

Our embedded input is first passed through a 2-layer BiLSTM. Due to its bidirectional nature, the BiLSTM doubles the embedding dimension by concatenating features from both the forward and backward passes. After the BiLSTM layer, we apply the padding mask to suppress outputs corresponding to padded positions, preventing invalid time steps from influencing downstream computations. To reduce this expanded dimension back to our desired size while still retaining the bidirectional information, we employ a convolutional projection block, which we refer to as the CNN Projection in Fig. 5 to distinguish it from the CNN Module described in Sect. 3.5, as this block acts primarily as a channel reducer, projecting the $2d_{\mathrm{emb}} = 128$ BiLSTM features down to $d_{\mathrm{emb}} = 64$. We found this to be more effective than simply halving the hidden dimension of the BiLSTM in each direction. Preliminary experiments demonstrated that the post BiLSTM convolutional projection approach yielded higher classification accuracy, likely because the learned kernels better preserve the salient features extracted by the BiLSTM passes. While this approach increases the computational complexity and parameter count, the gain in predictive performance justifies the additional overhead.

The convolutional projection block is composed of three sequential 1-D convolutions with Group Normalization (`GroupNorm`, Wu & He 2018) and ReLU activation: a pointwise convolution with 128 input channels and 256 output channels, a kernel size 3 convolution with 256 input channels and 256 output channels, and another pointwise convolution with 256 input channels and 64 output channels that returns the tensor to the set dimensionality. Two of the three layers are pointwise and act purely along the channel dimension, while the middle kernel-3 layer additionally mixes information across neighboring time steps for further local temporal context. We use `GroupNorm` rather than Batch Normalization (`BatchNorm`, Ioffe & Szegedy 2015) to improve stability with variable-length sequences and smaller batch sizes, as BatchNorm statistics can be unreliable when sequences contain varying amounts of padding. The output forms a residual connection with the input to the BiLSTM which is then normalized and passed into the Transformer. Beyond extracting useful features, the residual block serves as an additional form of positional encoding, enhancing the Transformer's ability to model temporal dependencies.

3.4. *Transformer Encoder*

In the Transformer layer, we use the positional encoding described in Sect. 2.1 and Eqs. 2-3, 8-head attention and replace the standard position-wise feed-forward network with a convolutional feed-forward module. This substitution allows the feed-forward stage to incorporate local temporal context from neighboring time steps, rather than processing each position independently.

The attention mechanism incorporates the padding mask to ensure that attention is restricted to valid (non-padded) positions only. This is achieved by setting the attention scores for masked positions to negative infinity before the softmax operation, effectively zeroing their attention weights. The residual connection of the transformer encoder is then input to our CNN module.

3.5. *CNN Module*

The CNN module (Fig. 7) applies its convolutions along the time dimension, with the multi-scale kernels (sizes 3, 7, 15, and 111) capturing temporal patterns at progressively larger receptive fields. It passes the Transformer output through a pointwise convolution with 64 input channels and 256 output channels, followed by a Gated Linear Unit (GLU) which halves the channels to 128. This is followed by 4 convolutions of kernel sizes 3, 7, 15, and 111, each with 128 input and output channels and followed by a `GroupNorm` and Sigmoid Linear Unit (SiLU) activation function. This range of different kernel sizes allows the model to capture both local and global trends from the Transformer's output. The specific combination of kernels was selected through an iterative optimization process on the validation set. As in the CNN Projection block, `GroupNorm` (with a single group) is used throughout the module to maintain consistent normalization behavior regardless of the padding ratio within each batch. Lastly, another pointwise convolution is performed with 128 input channels and 64 output channels to return to the set dimensionality. The output forms a residual connection with the original CNN input, which is then normalized.

3.6. *Output Layer*

The final output probabilities are obtained by applying global average pooling across the time dimension. Because we work with variable-length sequences, we use masked averaging, that is, summing only over valid (non-padded) positions and dividing by the count of valid time steps. This ensures that padding tokens do not influence the learned representations. Formally, given the sequence output $\mathbf{H} \in \mathbb{R}^{T\times d}$, where T is the number of time steps and d is the embedding dimension, and a binary mask $\mathbf{m} \in \{0,1\}^T$ indicating the valid positions, the pooled representation is computed as:

$$\mathbf{z} = \frac{\sum_{t=1}^{T} m_t \cdot \mathbf{h}_t}{\max\left(1, \sum_{t=1}^{T} m_t\right)} \tag{6}$$

where h_t is the t-th row of $\mathbf{H}$. The pooled embedding is then passed through a 3-layer MLP for classification. The MLP consists of a linear layer projecting from 64 to 128 dimensions, followed by Layer Normalization (`LayerNorm`, Ba et al. 2016), SiLU activation, and dropout (p=0.2). A second linear layer reduces the dimension from 128 to 32, followed by SiLU activation and dropout (p=0.2). Finally, a linear layer maps from 32 dimensions to 8 class logits and is followed by a softmax function to produce output probabilities.

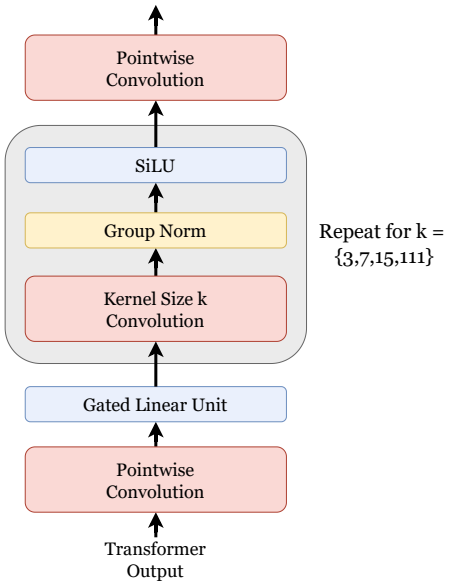

Figure 7. The CNN module.

When performing inference, Monte-Carlo dropout (Gal & Ghahramani 2015) is applied to calibrate probabilities and estimate predictive uncertainty. This consists of keeping dropout active during inference with a probability of 0.2 and performing 20 forward passes, taking the mean of these outputs as the final predicted probability distribution.

4. TRAINING DATA

Our training data consists of two labeled datasets: *Kepler* light curves (Sect. 4.1) and TESS QLP light curves (Sect. 4.2).

4.1. *Kepler*

We first validate the performance of our architecture on light curves from the *Kepler* mission. We use the labeled benchmark dataset from Audenaert et al. (2021), which consists of the following eight classes: (1) aperiodic variables (APERIODIC), (2) constant variables (CONSTANT), (3) contact binaries and rotational variables (CONTACT_ROT), (4) δ Scuti and β Cephei stars (DSCT_BCEP), (5) eclipsing binaries and transit events (ECLIPSE), (6) γ Doradus and Slowly Pulsat-

ing B stars (GDOR_SPB), (7) RR Lyrae and Cepheid variables (RRLYR_CEPH), and (8) solar-like pulsators (SOLARLIKE). The detailed class descriptions are provided in Audenaert et al. (2021).

4.2. *TESS*

We cross-match the *Kepler* training set (Audenaert et al. 2021) with TESS based on the TESS Input Catalog (Stassun et al. 2018). In order to increase the number of examples in challenging and smaller classes, we extend the RRLYR_CEPH class with additional Cepheids and RR Lyraes from (Ripepi et al. 2023; Clementini et al. 2023), the DSCT_BCEP, GDOR_SPB and CONTACT_ROT classes with the targets identified by Skarka et al. (2022); Skarka & Henzl (2024). These samples were manually cleaned from ambiguous cases. We then retrieved the available MIT Quick-look Pipeline (QLP, Huang et al. 2020a,b; Kunimoto et al. 2021, 2022) light curves for the constructed catalog in sectors 14, 15 and 26 (*Kepler FoV*), and perform visual inspections based on the light curves and periodograms and remove those light curves where no clear signal is found in the TESS QLP. Overall, there is a significant reduction in the number of unique targets because many of the light curves are dominated by noise and systematic properties that hide the astrophysical signatures. However, this is partially compensated by the inclusion of multiple sectors of data for the same target star. It is challenging to detect the oscillation and granulation patterns for solar-like oscillators based on single sector light curves, resulting in an overall reduction of the class size. Because of the large number of light curves dominated by systematic and instrumental trends, and in line with the findings from Audenaert et al. (2021); Tey et al. (2023), we also add an INSTRUMENT/JUNK class to minimize confusion with astrophysical classes, and essentially replace the CONSTANT class with it because it consisted of simulated light curves. We populated the INSTRUMENT/JUNK class by selecting light curves from initial classification results that exhibited large recurring systematic trends or a lack of variability. The final training set is shown in Table 1.

4.3. *Light curve preprocessing*

We preprocess the light curves before feeding them to the model to remove noise and systematic trends in order to optimize performance, in line with Hey & Aerts (2024) and Kliapets et al. (2025). We first remove the time steps and flux values flagged by QLP, remove NaN values and outlier values that deviate from the median by more than ten times the standard deviation. Subsequently, we run a 1-D Gaussian filter ($\sigma = 61$) and subtract it from the light curve. This is because a Gaussian filter with high sigma value represents long range trends that are often present in TESS light curves but irrelevant to the stellar variability pattern. With this filter, the longest period that can pass through the Gaussian filter is 7.665 d given the TESS sampling frequency $f_s = 48\ d^{-1}$ (0.02 d) in the nominal mission. The relation between the standard deviation in time (σ_t) and frequency (σ_f) domains is then $\sigma_f = \frac{0.02}{2\pi\sigma_t}$. We note that the Gaussian filter could remove long-term stellar variability trends such as the year-long beating periods in g-mode pulsators previously found in Kepler by Van Beeck et al. (2021). Given those are not the primary focus of our research this is not an issue. Lastly, we shift the time values to start at zero in order to work better with the Transformer's positional encoding and standardize the flux values. For *Kepler* light curves, we apply only standardization, as the higher data quality requires less pre-processing compared to TESS.

5. TRAINING

We train our model using a batch size of 128 light curves. We use the AdamW optimizer (Loshchilov & Hutter 2017) with a learning rate (γ) of 10^{-4} for the ASTRAFier blocks and 10^{-3} for the MLP classification head, a weight decay coefficient (λ) of 1×10^{-5}, first and second moment decay rates (β_1 and β_2) of 0.9 and 0.95, respectively, and a numerical stability term (ϵ) of 10^{-8}. We use the AdamW optimizer due to its decoupled weight decay, which applies weight decay directly to the weights independent of gradient update. This leads to better convergence and regularization. To add further regularization, we use dropout layers (Srivastava et al. 2014) with dropout probability of 0.2 throughout our model. We employ class-weighted cross-entropy loss with weights inversely proportional to class frequency ($w_c = N_{\text{total}}/N_c$) to mitigate the effects of class imbalance. We split our data into 80% training and 20% holdout sets, stratified by class and split at the TIC level to ensure no target appears in both sets. During training, we further partition 10% of the training set to obtain a validation set. We select the best-performing model based on validation accuracy and report its results on the holdout set.

5.1. *Computational complexity*

Our final model contains 8.8 million parameters with a size of 35 MB. We train on 2 × NVIDIA H200 GPUs, with each epoch completing in approximately 1.5 minutes for 12,038 training samples. On the two H200

Table 1. The number of light curves for each class across our Kepler and TESS QLP datasets used in this work.

Dataset	APERIODIC	CONSTANT	CONTACT_ROT	DSCT_BCEP	ECLIPSE	GDOR_SPB	INST./JUNK	RRLYR_CEPH	SOLARLIKE	Total
Kepler	830	1000	2260	772	974	630	0	62	1800	8328
TESS QLP	1197	0	1489	1981	851	1085	1499	289	952	9343

GPUs, the classification of 1 million light curves with 20 forward passes each (for Monte Carlo dropout, see Sect. 3.6) completes in approximately 8 hours.

6. RESULTS

We discuss the results of our model on the *Kepler* and TESS datasets, validate our architectural choices and show the model's ability to accurately classify light curves.

We evaluate our architecture in three stages. We first train and test on *Kepler* alone to benchmark against prior work (Sect. 6.1), then train and test on TESS alone to assess performance with our refined class structure (Sect. 6.2), and finally train on combined *Kepler* and TESS data but evaluating on the TESS holdout set, to produce our final deployment model (Sect. 6.3).

6.1. *Kepler*

Training on our *Kepler* dataset, we achieve a classification accuracy of 94.26% on the holdout set. The confusion matrix for the holdout set is shown in Fig. 8, and the recall, precision, and F1 scores[1] for each class are shown in Table 2. The final estimates are computed by averaging over the scores of the eight classes.

The overall accuracy is comparable to that of Audenaert et al. (2021), who achieved an accuracy of 94.90% on a different holdout set. Comparing the performance on a class-by-class basis, our model fails to correctly identify stars from the GDOR_SPB class more often than in Audenaert et al. (2021), with most of the confusion being with the CONTACT_ROT class, a well-known challenge in variability classification (e.g., Audenaert et al. 2021; Barbara et al. 2022; Hey & Aerts 2024). Our model also fails to correctly identify the RRLYR_CEPH class more often with most of the confusion being again with the CONTACT_ROT class. However, there are only 12 RRLYR_CEPH stars in our holdout set, making it less reliable. Our model is better able to identify eclipses, achieving a recall of 100%. The remaining classes are all within 1% when comparing results. It should be noted again that the results from Audenaert et al. (2021) are on a different training and holdout set split, so comparisons should be interpreted with caution.

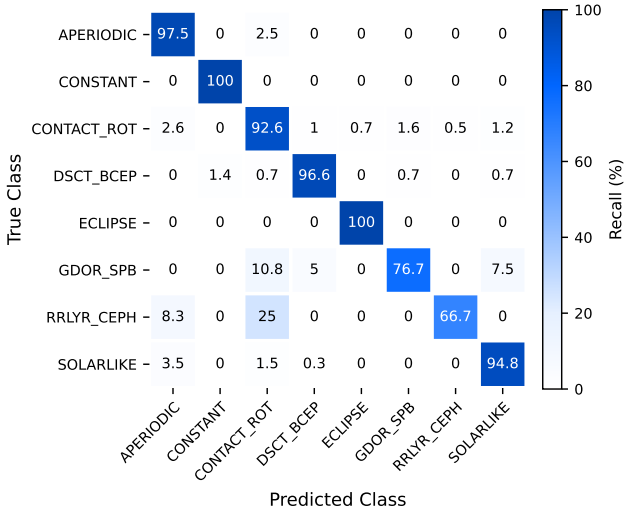

Figure 8. The confusion matrix on the *Kepler* holdout set for the model trained only on *Kepler* data.

Table 2. Performance of the model trained on *Kepler* data, on the Kepler holdout set (in %).

Class	Recall	Precision	F1
APERIODIC	97.47(154/158)	86.52(154/178)	91.67
CONSTANT	100.00(190/190)	98.96(190/192)	99.48
CONTACT_ROT	92.56(398/430)	93.87(398/424)	93.21
DSCT_BCEP	96.58(141/146)	92.76(141/152)	94.63
ECLIPSE	100.00(185/185)	98.40(185/188)	99.20
GDOR_SPB	76.67(92/120)	92.00(92/100)	83.64
RRLYR_CEPH	66.67(8/12)	80.00(8/10)	72.73
SOLARLIKE	94.75(325/343)	95.58(325/340)	95.17
Total	90.59	92.26	91.21
Overall Accuracy		94.26	

6.2. *TESS*

Using the refined class structure described in Sect. 4.2, we first evaluate our model trained exclusively on the TESS dataset. On the TESS holdout set, we achieve a

[1] We calculate these metrics as follows: recall $= \frac{TP}{TP+FN}$, precision $= \frac{TP}{TP+FP}$, and F1 $= \frac{2*recall*precision}{recall+precision}$ where TP is the number of true positives for a class, FP the number of false positives, and FN the number of false negatives.

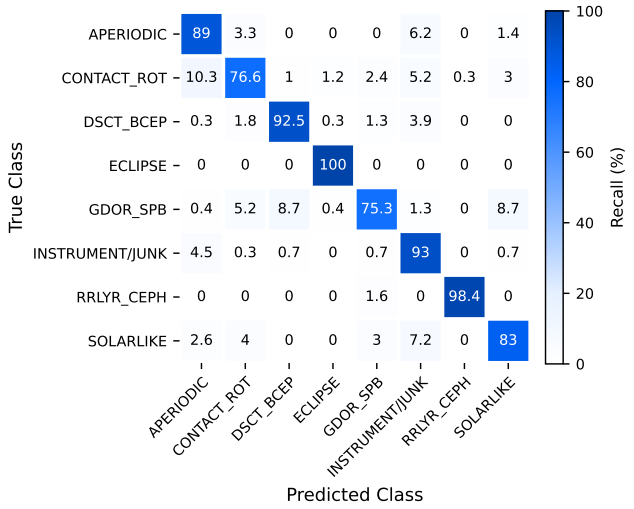

Figure 9. The confusion matrix on the TESS holdout set for the model trained only on TESS data.

classification accuracy of 87.09%. The confusion matrix is shown in Fig. 9, and per-class metrics are reported in Table 3. The most significant source of confusion is between the CONTACT_ROT and APERIODIC classes, where 10.3% of CONTACT_ROT stars are misclassified as APERIODIC. This is likely due to rotational variables whose periods exceed or are comparable to the 27.4-day sector baseline, resulting in light curves that lack clear periodicity and are thus difficult to distinguish from aperiodic variability.

The introduction of the INSTRUMENT/JUNK class proves effective, capturing instrumental and non-variable artifacts with 93.13% accuracy while limiting contamination of the astrophysical classes. The DSCT_BCEP and RRLYR_CEPH classes also perform well, with accuracies of 92.49% and 98.44%, respectively, although for RRLYR_CEPH we only have 64 light curves across 49 targets.

We achieve 100% accuracy on our ECLIPSE class; however, we do not expect this to hold in deployment. The holdout set contains only 153 light curves from 100 unique targets, and examining the prediction probabilities, 142 of the 153 eclipses in the holdout set receive a confidence above 95% (median 95.71%), indicating that these are predominantly unambiguous eclipsing signals. We expect misclassifications to occur on the full dataset, where less obvious or noisier eclipses will present more challenging cases.

6.3. *TESS and Kepler*

Using the refined class structure, we trained our final deployment model on a combined dataset of TESS (Sect. 4.2) and Kepler (Sect. 4.1) light curves, removing any Kepler targets already present in the TESS set to avoid data leakage. Since our goal is deployment on TESS, we evaluate on the TESS holdout set.

Table 3. Performance of the model trained on TESS data, on the TESS holdout set (in %).

Class	Recall	Precision	F1
APERIODIC	89.10(188/211)	77.69(188/242)	83.00
CONTACT_ROT	76.60(252/329)	87.80(252/287)	81.82
DSCT_BCEP	92.49(357/386)	93.46(357/382)	92.97
ECLIPSE	100.00(153/153)	96.23(153/159)	98.08
GDOR_SPB	75.32(174/231)	88.78(174/196)	81.50
INSTRUMENT/JUNK	93.13(271/291)	81.38(271/333)	86.86
RRLYR_CEPH	98.44(63/64)	98.44(63/64)	98.44
SOLARLIKE	82.99(161/194)	82.14(161/196)	82.56
Total	88.51	88.24	88.15
Overall Accuracy		87.09	

On the TESS holdout set, we achieve a classification accuracy of 88.22%. The confusion matrix is shown in Fig. 10, the UMAP visualization of holdout set embeddings in Fig. 11, and per-class metrics in Table 4.

As in Sect. 6.2, the ECLIPSE class achieves perfect recall; we again attribute this to the holdout set containing predominantly unambiguous eclipsing signals rather than expecting this to generalize to deployment, and noting the limited holdout set size of 153 light curves across 100 targets. The RRLYR_CEPH class also achieves 100% recall, though with only 64 light curves from 49 unique targets in the holdout set, this should be interpreted with caution.

As shown in Table 5, we also demonstrate increased performance when adding *Kepler* light curves to our training set, with F1 scores improving for seven of eight classes and a macro-averaged F1 gain of +1.04. The most notable improvements are in GDOR_SPB (+3.11) and SOLARLIKE (+2.81). The only notable decrease is RRLYR_CEPH (−2.20), which we attribute to the small class size making precision sensitive to even a few additional false positives. These gains demonstrate that our model scales effectively with training set size, suggesting further improvements are achievable as more labeled data becomes available.

We additionally validate our architectural choices through an ablation study (Table 6), in which we remove one of the three core modules (BiLSTM, Attention, or CNN) from ASTRAFier and retrain; the full architecture outperforms all reduced variants, confirming that each component contributes meaningfully to classification performance.

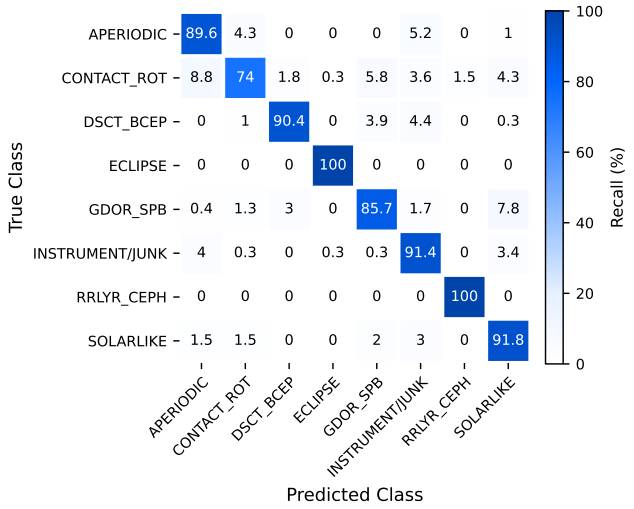

Figure 10. The confusion matrix on the TESS holdout set for the final model trained on both *Kepler* and TESS data.

Table 4. Performance of the final model trained on *Kepler* and TESS data, on the TESS holdout set (in %).

Class	Recall	Precision	F1
APERIODIC	89.57(189/211)	80.77(189/234)	84.95
CONTACT_ROT	73.86(243/329)	92.40(243/263)	82.10
DSCT_BCEP	90.41(349/386)	96.41(349/362)	93.31
ECLIPSE	100.00(153/153)	98.71(153/155)	99.35
GDOR_SPB	85.71(198/231)	83.54(198/237)	84.61
INSTRUMENT/JUNK	91.41(266/291)	84.18(266/316)	87.64
RRLYR_CEPH	100.00(64/64)	92.75(64/69)	96.24
SOLARLIKE	91.75(178/194)	79.82(178/223)	85.37
Total	90.34	88.57	89.20
Overall Accuracy		88.22	

Table 5. Comparison on our TESS holdout set of F1 scores between the TESS-only model and the final model trained on combined TESS and *Kepler* data.

Class	TESS-only F1	TESS+*Kepler* F1	Change
APERIODIC	83.00	84.95	+1.95
CONTACT_ROT	81.82	82.10	+0.28
DSCT_BCEP	92.97	93.31	+0.34
ECLIPSE	98.08	99.35	+1.27
GDOR_SPB	81.50	84.61	+3.11
INSTRUMENT/JUNK	86.86	87.64	+0.78
RRLYR_CEPH	98.44	96.24	−2.20
SOLARLIKE	82.56	85.37	+2.81
Macro F1	88.15	89.20	+1.04

Table 6. Ablation study on the combined TESS and Kepler holdout set. Each row removes one core module from ASTRAFier.

Model	Accuracy	Recall	Precision	F1
ASTRAFier	88.22	90.34	88.57	89.20
Attention + CNN	87.14	88.76	86.73	87.44
BiLSTM + Attention	86.66	88.24	87.30	87.66
BiLSTM + CNN	85.48	87.37	85.84	86.35

NOTE—For this experiment, we remove one module (LSTM, Attention, CNN) from our ASTRAFier model, keeping everything else the same. When we remove LSTM, we also remove the 3-layer CNN in its residual block. Recall, precision, and F1 are the macro-averaged score across all classes.

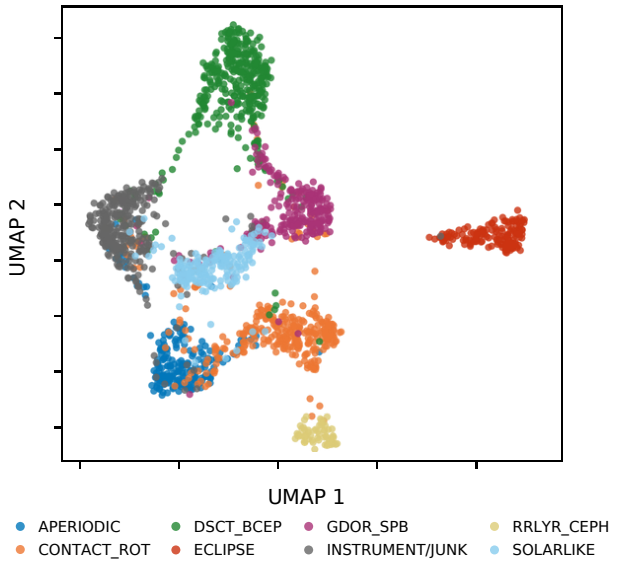

Figure 11. The UMAP reduction of the data points in our final QLP holdout set extracted before the final MLP layer.

We can visualize how well the model separates different classes by extracting the embeddings from before the final MLP layers and plotting them in 2 dimensions using UMAP (McInnes et al. 2018). Examining the plot, there is noticeable overlap between the DSCT_BCEP and GDOR_SPB classes. These could be hybrid pulsating stars that exhibit both p and g modes (e.g., Fritzewski et al. 2025a; Kliapets et al. 2025). We can also see confusion between the CONTACT_ROT and APERIODIC classes, as well as the GDOR_SPB and SOLARLIKE classes; which is consistent with the confusion matrix in Fig. 10.

7. DEPLOYING THE CLASSIFIER

We deploy our final model trained on *Kepler* and TESS data on all $\sim$ 2.8 million QLP light curves observed in TESS sectors 14, 15 and 26. In general, we find that the accuracy scores are in line with the reported testing scores in Sect. 6. However, because a training set is never a perfect representation of reality (e.g., small class sizes, varying systematics,...), there are always differences between testing and deployment performance on the full dataset.

We therefore evaluate the performance of our variability classification architecture during deployment by analyzing the astrophysical properties exhibited by subpopulations assigned a certain class, and support this with detailed inspections of their light curves and amplitude spectra.

In Fig. 12, we show the distributions of effective temperature ($T_{\rm eff}$ from *Gaia* DR3 data, top row), dominant variability f_1 (middle row), and its amplitude A_1 (bottom row) — on which the classifier had no prior information. We only plotted targets that received final scores above 0.5 per class; we revisit this later in this Section. We note that the frequencies and amplitudes of the OBAF-type pulsators are mostly in line with those in Hey & Aerts (2024) and Aerts et al. (2025). One notable exception is the distribution of A_1 for RRLYR_CEPH class, which for the unlabeled TESS data is shifted much further to the left and peaks at very low amplitudes despite high performance demonstrated in Sect. 6.3. This could be explained by this class being the least represented in the training data, leading to issues in successfully generalizing to unseen data while differentiating this class from rotational variables. This was supported by our manual inspection of random light curves that received high probabilities for this class. The rotation periods (inverses of f_1) for the CONTACT_ROT class are comparable to the ones reported by Colman et al. (2024) and are biased towards short periods, as expected from the TESS data. Other potential misclassifications revealed by the distributions are the confusion of solar-like oscillators with g-mode pulsators where $T_{\rm eff} < 6500$ K, as well as stars in the first peak of the bimodal A_1 distribution — typically slower-rotating g-mode pulsators — and which is higher in the unlabeled data than the labeled set, which coincides with the A_1 distribution peak for solar-like oscillators, consistent with Fig. 10.

The latter is further supported by the analysis of amplitude spectra. In Fig. 13, we show the stacked periodograms for stars labelled as g- or p-mode pulsators by the classifier. For g-mode pulsators, similar to Li et al. (2020) and Hey & Aerts (2024), we see a main ridge on the stacked amplitude spectra (top plot, in period), mostly populated by the prograde dipole ($(l, m) = (1, 1)$) mode (Aerts & Tkachenko 2024). The secondary lower ridge is likely associated with a lower-amplitude with $l = 2$ or a harmonic of a dominant mode (Hey & Aerts 2024). Some targets also show potential r modes similar to those from Li et al. (2020). Stars immediately below the main ridge are once again likely misclassified solar-like oscillators. On the bottom of the plot, we see stars with clear harmonic behaviors, likely rotational variables or eclipsing binaries. A clear vertical ridge at 1 d^{-1} is likely a light curve systematic. For p-mode pulsators (bottom plot, in frequency), no structures other than the dominant mode can be seen, similar to what was found by Fritzewski et al. (2025b).

Finally, we also investigated the position of candidate pulsators on the Hertzsprung–Russell (HR) diagram. On the top panel of Fig. 14, we show the positions of 5% randomly-sampled light curves with probabilities higher than 0.5 on the HR diagram (for APERIODIC, ECLIPSE, and INSTRUMENT/JUNK only 1% is plotted for visibility), which we found to be mostly in line with what expected from the respective types of stars (Aerts 2021). We note that stars classified as SOLAR-LIKE are found both on the Main Sequence (MS) and in the Giant Branch despite the differences in both excitation mechanisms and typical amplitudes. We manually inspected some light curves for this class in both regions of the HR diagram, which revealed that they share similar light curve and periodogram structures, as expected from stars being put in the same class. We found that some of them lack power excess in frequency ranges expected from either solar-like stars on the MS or solar-like oscillators (pulsating red giants). Particularly, stars labelled SOLARLIKE on the MS where most of the power is concentrated in frequencies below 0.5 d^{-1}, are likely misclassifications. This suggests that automatic detection of solar-like oscillators in TESS data is challenging.

The bottom panel shows candidate p- and g-mode pulsators (each point is a normalized probability distribution of a target assigned DSCT_BCEP and GDOR_SPB labels), which reveals a number of stars populating space in the gap between β Cep / δ Sct stars and SPB / γ Dor stars, similar to De Ridder et al. (2023), Hey & Aerts (2024), Mombarg et al. (2024), Aerts et al. (2025), and Kliapets et al. (2025). Previous studies suggested that these stars could potentially appear cooler due to rotating spots (De Ridder et al. 2023). These candidate pulsators are excellent targets for more detailed studies challenging the theoretical bounds of instability strips. We additionally note that a number of stars with high probabilities of being g-mode (and to the lesser extent, p-mode) pulsators, are found in the Red Giant Branch. These are potentially misclassified red giants (solar-like

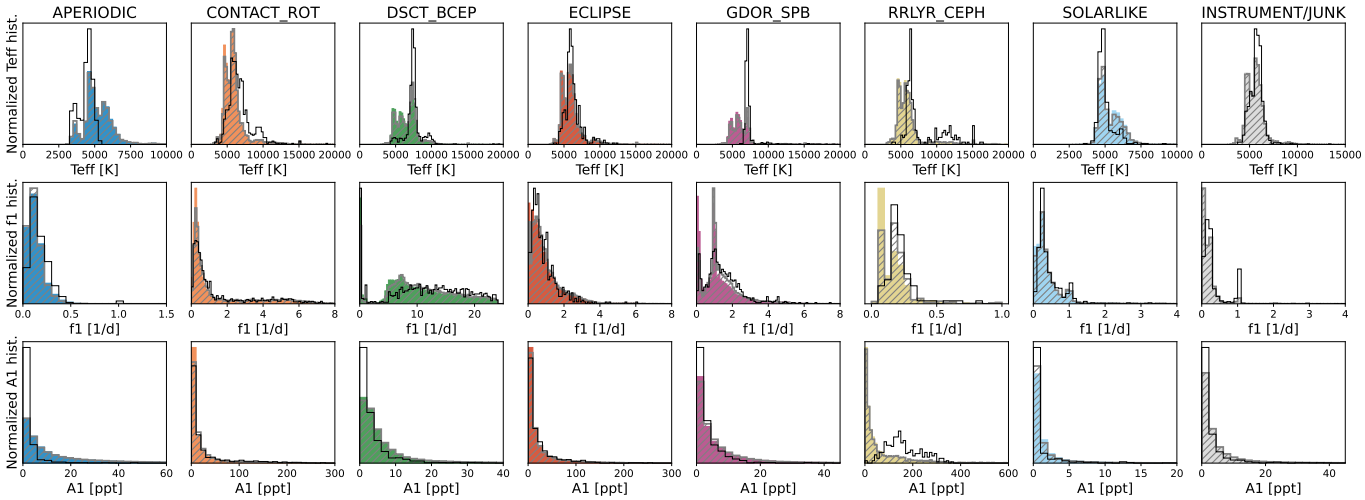

Figure 12. Normalized distributions of effective temperatures (top), dominant variability (middle), and its amplitude (bottom) of the labeled set (black outline) and classified targets from TESS Sectors 14, 15, and 26 with probabilities higher than 0.5 (color) and 0.8 (hash). If a star had more than one light curve in the *Kepler* field of view, only the one with the higher probability was plotted. Distributions have been clipped on the right for visibility at different values for each class.

oscillators), which is common for automated pipelines. We tested this hypothesis by inspecting some of these light curves and found that stars labelled GDOR_SPB fall into one of the two categories: (i) true g-mode pulsators with a wrong $T_{\rm eff}$; or (ii) predominantly misclassified red giants or, rarer, rotational variables. Stars labelled DSCT_BCEP are practically entirely true p-mode pulsators with a wrong $T_{\rm eff}$ and some notable instrumental power excess in the low-frequency regime.

The potential misclassifications revealed by these analyses suggest that using a probabilistic cut-off of 0.5 is too optimistic. Based on visual inspections, we therefore suggest using a threshold per class of 0.75-0.8, depending on accuracy requirements. We do note that even for higher probability bins, we still see some of the discussed confusion happening, with the biggest difference between the confusion matrix in Fig. 10 and the deployment results happening for the RRLYR_CEPH class because of the limited training set.

8. DISCUSSION AND CONCLUSIONS

In this work, we introduced the ASTRAFier stellar variability classification model. The architecture combines BiLSTM, Attention, and CNN components, which each play an important and complementary role in processing the light curves. The model works directly on the time series, eliminating the need for feature extraction. We have demonstrated the effectiveness of the model in classifying variability, achieving 94.26% classification accuracy on *Kepler* and 88.22% on TESS data. The classification performance is in line with Audenaert et al. (2021) but comes with a much lower computational complexity and model complexity. The rapid classification inference time allows us to more easily classify millions of TESS light curves, while the simpler model architecture allows for better software maintenance. Our deep learning architecture also offers more flexibility for detecting variability classes that are currently not included in our classification scheme, because it does not rely on specific features but can just be retrained to look for new types of patterns.

We found that the performance of our model clearly scales with the size of the training set. Given that Transformers inherently operate within a large hypothesis space, they are known to be particularly data-hungry when trained from scratch, meaning they require large amounts of training data. Although we attempt to mitigate this challenge through the inclusion of LSTM and CNN layers, as well as various regularization techniques, the model remains susceptible to overfitting due to the relatively small size of our labeled training set.

In particular, we can increase the size of our training set by including the data from the TESS extended missions, as we currently only included primary mission data. However, the shorter cadence of the extended mission light curves leads to longer sequences with more time-steps. While this could lead to a more precise light curve with more distinct variability, especially for p-mode pulsators, it is possible that the longer sequences make it more difficult for the model to learn long-range dependencies and increase computational costs. This could potentially be addressed by downsampling the data. For example, Kliapets et al. (2025) found that the recovery of dominant and secondary variability from Kepler in TESS is better in downsampled extended mission

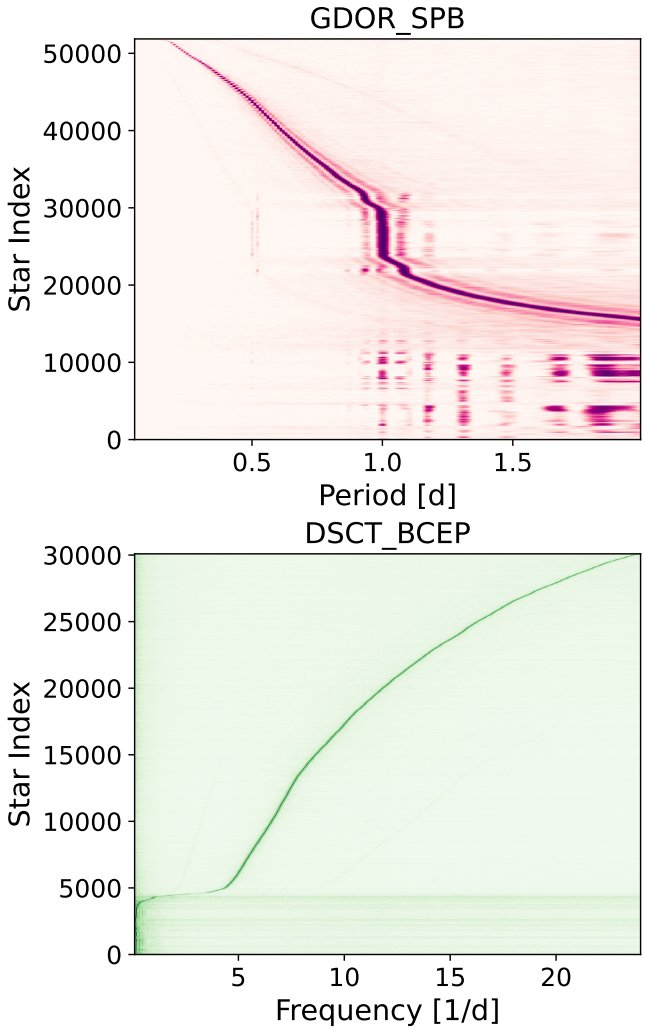

Figure 13. Stacked amplitude spectra of candidate g-mode (top, in period) and p-mode pulsators (bottom, in frequency), for which the prediction probability is higher than 0.5. Stars of each of the two classes are sorted by the dominant variability.

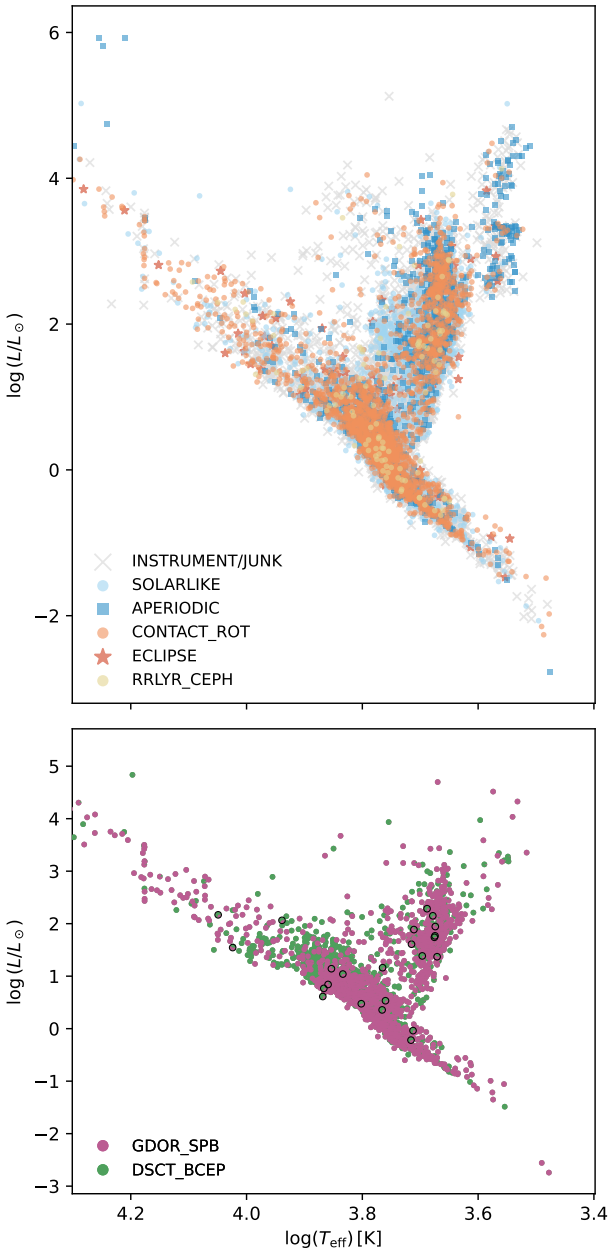

Figure 14. HR Diagram of randomly-sampled high-probability candidate light curves for all classes except DSCT_BCEP and GDOR_SPB (top panel) and, separately, only DSCT_BCEP and GDOR_SPB with normalized probabilities for the two classes (bottom panel). Black outlines mark stars with the secondary class having a normalized probability higher than 0.2 — potentially hybrid pulsators. A vertical line at 15,000 K is a *Gaia* DR3 grid systematic.

data than for the nominal mission data with the same cadence.

We demonstrated the current computational scalability of our approach by classifying $\sim$ 2.8 million light curves from TESS sectors 14, 15, and 26, constructing a comprehensive catalog of candidate variable stars in these sectors. The code and trained model are publicly available[2]. We are now working on extending our methodology to run on the TESS-Gaia light curves (TGLC, Han & Brandt 2023), of which the aperture light curve methodology has been incorporated in the QLP pipeline since sector 94[3] (Petitpas et al. 2026). In particular, we are classifying all TGLC light curves in

[2] https://github.com/jeraud/TESS-Transformer

[3] https://tess.mit.edu/qlp/

the PLATO Field-of-View in order to construct a variability catalog for the PLATO Complementary Science Program (Kliapets et al, in prep.).

Lastly, the scaling of data size and performance cannot only be addressed by increasing the size of the labeled training set, but can also be tackled by moving to a self-supervised learning scheme that can take advantage of unlabeled data (see e.g., Parker et al. 2024; Audenaert 2025, for an explanation). ASTRAFier is being used to create a foundation model (see e.g., Bommasani et al. 2021, for an explanation) for TESS (Audenaert et al. 2025) that can be used for a much wider variety of downstream tasks (clustering, anomaly detection, parameter estimation,...), where we are additionally incorporating the ability to remove instrumental and systematic effects (Audenaert et al. 2025; Mercader-Perez et al. 2026).

Funding for the TESS, Kepler and K2 mission is provided by NASA's Science Mission Directorate. The research leading to these results has received funding from MIT's Undergraduate Research Opportunities Program (UROP), the BELgian federal Science Policy Office (BELSPO) through the PRODEX grant for PLATO. MK acknowledges The Kavli Foundation for their financial support in the framework of the Kavli Scholarship given to MK from 25/9/2023-24/9/2025, including facilitation of MK's research visit to the MIT Kavli Institute for Astrophysics and Space Research in the fall of 2025 (hosts: JA and GRR). The authors acknowledge the MIT Office of Research Computing and Data (ORCD) for providing high performance computing resources. The authors would like to acknowledge the valuable contributions and feedback provided by members of the TESS Asteroseismic Science Consortium.

Software: astropy (Astropy Collaboration et al. 2013, 2018), Lightkurve (Lightkurve Collaboration et al. 2018), Matplotlib (Hunter 2007), NumPy (Harris et al. 2020), pandas (McKinney 2010), PyTorch (Paszke et al. 2019), PyTorch Lightning (Falcon & The PyTorch Lightning team 2019), scikit-learn (Pedregosa et al. 2011), SciPy (Virtanen et al. 2020), UMAP (McInnes et al. 2018)

REFERENCES

Aerts, C. 2021, Reviews of Modern Physics, 93, 015001, doi: 10.1103/RevModPhys.93.015001

Aerts, C., Christensen-Dalsgaard, J., & Kurtz, D. W. 2010, Asteroseismology, doi: 10.1007/978-1-4020-5803-5

Aerts, C., & Tkachenko, A. 2024, Astronomy & Astrophysics, 692, R1

Aerts, C., Van Reeth, T., Mombarg, J. S., & Hey, D. 2025, Astronomy & Astrophysics, 695, A214

Armstrong, D. J., Kirk, J., Lam, K. W. F., et al. 2016, MNRAS, 456, 2260, doi: 10.1093/mnras/stv2836

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33, doi: 10.1051/0004-6361/201322068

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123, doi: 10.3847/1538-3881/aabc4f

Audenaert, J. 2025, Ap&SS, 370, 72, doi: 10.1007/s10509-025-04460-5

Audenaert, J., Muthukrishna, D., Gregory, P., Hogg, D., & Villar, V. A. 2025, ICML 2025 Workshop on Foundation Models for Structured Data. https://arxiv.org/abs/2507.05333

Audenaert, J., & Tkachenko, A. 2022, A&A, 666, A76, doi: 10.1051/0004-6361/202243469

Audenaert, J., Kuszlewicz, J. S., Handberg, R., et al. 2021, AJ, 162, 209, doi: 10.3847/1538-3881/ac166a

Ba, J. L., Kiros, J. R., & Hinton, G. E. 2016, Layer Normalization. https://arxiv.org/abs/1607.06450

Barbara, N. H., Bedding, T. R., Fulcher, B. D., Murphy, S. J., & Van Reeth, T. 2022, MNRAS, 514, 2793, doi: 10.1093/mnras/stac1515

Becker, I., Protopapas, P., Catelan, M., & Pichara, K. 2025, Multiband Embeddings of Light Curves. https://arxiv.org/abs/2501.12499

Blomme, J., Sarro, L. M., O'Donovan, F. T., et al. 2011, MNRAS, 418, 96, doi: 10.1111/j.1365-2966.2011.19466.x

Bommasani, R., Hudson, D. A., Adeli, E., et al. 2021, arXiv preprint arXiv:2108.07258, arXiv:2108.07258, doi: 10.48550/arXiv.2108.07258

Borucki, W. J., Koch, D., Basri, G., et al. 2010, Science, 327, 977, doi: 10.1126/science.1185402

Breiman, L. 2001, Machine Learning, 45, 5

Choi, J. Y., Espinoza-Rojas, F., Coppée, Q., & Hekker, S. 2025, A&A, 699, A180, doi: 10.1051/0004-6361/202555279

Clementini, G., Ripepi, V., Garofalo, A., et al. 2023, A&A, 674, A18, doi: 10.1051/0004-6361/202243964

Colman, I. L., Angus, R., David, T., et al. 2024, The Astronomical Journal, 167, 189

Cui, K., Armstrong, D. J., & Feng, F. 2024, ApJS, 274, 29, doi: 10.3847/1538-4365/ad62fd

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. 2016, arXiv e-prints, arXiv:1612.08083, doi: 10.48550/arXiv.1612.08083

De Ridder, J., Ripepi, V., Aerts, C., et al. 2023, Astronomy & Astrophysics, 674, A36

Debosscher, J., Sarro, L. M., Aerts, C., et al. 2007, A&A, 475, 1159, doi: 10.1051/0004-6361:20077638

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. 2018, arXiv e-prints, arXiv:1810.04805, doi: 10.48550/arXiv.1810.04805

Donoso-Oliva, C., Becker, I., Protopapas, P., et al. 2026, A&A, 707, A170, doi: 10.1051/0004-6361/202554026

—. 2023, A&A, 670, A54, doi: 10.1051/0004-6361/202243928

Eschen, Y. N. E., Bayliss, D., Wilson, T. G., et al. 2024, arXiv e-prints, arXiv:2409.13039, doi: 10.48550/arXiv.2409.13039

Falcon, W., & The PyTorch Lightning team. 2019, GitHub

Fetherolf, T., Pepper, J., Simpson, E., et al. 2023, ApJS, 268, 4, doi: 10.3847/1538-4365/acdee5

Foumani, N. M., Tan, C. W., Webb, G. I., & Salehi, M. 2023, arXiv e-prints, arXiv:2305.16642, doi: 10.48550/arXiv.2305.16642

Friedman, J. H. 2001, Annals of statistics, 1189

Fritzewski, D., Kemp, A., Li, G., & Aerts, C. 2025a, arXiv preprint arXiv:2512.09395

Fritzewski, D., Vanrespaille, M., Aerts, C., et al. 2025b, Astronomy & Astrophysics, 698, A253

Gal, Y., & Ghahramani, Z. 2015, arXiv e-prints, arXiv:1506.02142, doi: 10.48550/arXiv.1506.02142

Han, T., & Brandt, T. D. 2023, AJ, 165, 71, doi: 10.3847/1538-3881/acaaa7

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Nature, 585, 357, doi: 10.1038/s41586-020-2649-2

Hatt, E., Nielsen, M. B., Chaplin, W. J., et al. 2023, A&A, 669, A67, doi: 10.1051/0004-6361/202244579

Hey, D., & Aerts, C. 2024, A&A, 688, A93, doi: 10.1051/0004-6361/202450489

Hochreiter, S., & Schmidhuber, J. 1997, Neural Computation, 9, 1735, doi: 10.1162/neco.1997.9.8.1735

Hon, M., Stello, D., García, R. A., et al. 2019, MNRAS, 485, 5616, doi: 10.1093/mnras/stz622

Hon, M., Stello, D., & Yu, J. 2018a, MNRAS, 476, 3233, doi: 10.1093/mnras/sty483

Hon, M., Stello, D., & Zinn, J. C. 2018b, ApJ, 859, 64, doi: 10.3847/1538-4357/aabfdb

Howell, S. B., Sobeck, C., Haas, M., et al. 2014, PASP, 126, 398, doi: 10.1086/676406

Huang, C. X., Vanderburg, A., Pál, A., et al. 2020a, Research Notes of the American Astronomical Society, 4, 204, doi: 10.3847/2515-5172/abca2e

—. 2020b, Research Notes of the American Astronomical Society, 4, 206, doi: 10.3847/2515-5172/abca2d

Huber, D. 2025, arXiv e-prints, arXiv:2512.10002, doi: 10.48550/arXiv.2512.10002

Huijse, P., De Ridder, J., Eyer, L., et al. 2025, A&A, 701, A150, doi: 10.1051/0004-6361/202554025

Hunter, J. D. 2007, Computing in Science & Engineering, 9, 90, doi: 10.1109/MCSE.2007.55

IJspeert, L. W., Tkachenko, A., Johnston, C., & Aerts, C. 2024a, arXiv e-prints, arXiv:2409.20540, doi: 10.48550/arXiv.2409.20540

IJspeert, L. W., Tkachenko, A., Johnston, C., et al. 2021, A&A, 652, A120, doi: 10.1051/0004-6361/202141489

—. 2024b, A&A, 685, A62, doi: 10.1051/0004-6361/202349079

Ioffe, S., & Szegedy, C. 2015, in Proceedings of Machine Learning Research, Vol. 37, Proceedings of the 32nd International Conference on Machine Learning, ed. F. Bach & D. Blei (Lille, France: PMLR), 448–456. https://proceedings.mlr.press/v37/ioffe15.html

Jamal, S., & Bloom, J. S. 2020, ApJS, 250, 30, doi: 10.3847/1538-4365/aba8ff

Jannsen, N., Tkachenko, A., Royer, P., et al. 2025, A&A, 694, A185, doi: 10.1051/0004-6361/202452811

Kemp, A., Vrancken, J., Mombarg, J. S. G., et al. 2025, A&A, 704, A280, doi: 10.1051/0004-6361/202557362
Kim, D.-W., & Bailer-Jones, C. A. L. 2016, A&A, 587, A18, doi: 10.1051/0004-6361/201527188
Kliapets, M., Huijse, P., Tkachenko, A., et al. 2025, A&A, 703, A240, doi: 10.1051/0004-6361/202556079
Koch, D. G., Borucki, W. J., Basri, G., et al. 2010, ApJL, 713, L79, doi: 10.1088/2041-8205/713/2/L79
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, in Advances in Neural Information Processing Systems, ed. F. Pereira, C. Burges, L. Bottou, & K. Weinberger, Vol. 25 (Curran Associates, Inc.). https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
Kunimoto, M., Tey, E., Fong, W., et al. 2022, Research Notes of the American Astronomical Society, 6, 236, doi: 10.3847/2515-5172/aca158
Kunimoto, M., Huang, C., Tey, E., et al. 2021, Research Notes of the American Astronomical Society, 5, 234, doi: 10.3847/2515-5172/ac2ef0
Kurtz, D. W. 2022, ARA&A, 60, 31, doi: 10.1146/annurev-astro-052920-094232
LeCun, Y., Boser, B., Denker, J. S., et al. 1989, Neural Computation, 1, 541, doi: 10.1162/neco.1989.1.4.541
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, Proceedings of the IEEE, 86, 2278, doi: 10.1109/5.726791
Li, G., Van Reeth, T., Bedding, T. R., et al. 2020, Monthly Notices of the Royal Astronomical Society, 491, 3586
Lightkurve Collaboration, Cardoso, J. V. d. M., Hedges, C., et al. 2018, Lightkurve: Kepler and TESS time series analysis in Python, Astrophysics Source Code Library, record ascl:1812.013
Lomb, N. R. 1976, Ap&SS, 39, 447, doi: 10.1007/BF00648343
Loshchilov, I., & Hutter, F. 2017, arXiv e-prints, arXiv:1711.05101, doi: 10.48550/arXiv.1711.05101
McInnes, L., Healy, J., & Melville, J. 2018, arXiv e-prints, arXiv:1802.03426, doi: 10.48550/arXiv.1802.03426
McKinney, W. 2010, in Proceedings of the 9th Python in Science Conference, ed. S. van der Walt & J. Millman, 51–56
Mercader-Perez, P., Cuesta-Lazaro, C., Muthukrishna, D., et al. 2026, in ICLR 2026 Workshop on Foundation Models for Science: Real-World Impact and Science-First Design. https://openreview.net/forum?id=nebGk9bm3L
Mombarg, J. S., Aerts, C., Van Reeth, T., & Hey, D. 2024, Astronomy & Astrophysics, 691, A131
Moreno-Cartagena, D., Protopapas, P., Cabrera-Vives, G., et al. 2025, A&A, 703, A41, doi: 10.1051/0004-6361/202554289
Muthukrishna, D., Narayan, G., Mandel, K. S., Biswas, R., & Hložek, R. 2019, PASP, 131, 118002, doi: 10.1088/1538-3873/ab1609
Nascimbeni, V., Piotto, G., Cabrera, J., et al. 2025, A&A, 694, A313, doi: 10.1051/0004-6361/202452325
Naul, B., Bloom, J. S., Pérez, F., & van der Walt, S. 2018, Nature Astronomy, 2, 151, doi: 10.1038/s41550-017-0321-z
Nielsen, M. B., Hatt, E., Chaplin, W. J., Ball, W. H., & Davies, G. R. 2022, A&A, 663, A51, doi: 10.1051/0004-6361/202243064
Olmschenk, G., Barry, R. K., Ishitani Silva, S., et al. 2024, AJ, 168, 83, doi: 10.3847/1538-3881/ad55f1
Pan, J.-S., Ting, Y.-S., & Yu, J. 2024, MNRAS, 528, 5890, doi: 10.1093/mnras/stae068
Parker, L., Lanusse, F., Golkar, S., et al. 2024, MNRAS, 531, 4990, doi: 10.1093/mnras/stae1450
Paszke, A., Gross, S., Massa, F., et al. 2019, in Advances in Neural Information Processing Systems 32 (Curran Associates, Inc.), 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-libr pdf
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12, 2825
Petitpas, G., Haviland, J., Han, T., et al. 2026, arXiv e-prints, arXiv:2603.22236. https://arxiv.org/abs/2603.22236
Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. 2018
Ranaivomanana, P., Uzundag, M., Johnston, C., et al. 2025, A&A, 693, A268, doi: 10.1051/0004-6361/202452429
Ranjbar, M., & Rahimzadeh, M. 2024, arXiv e-prints, arXiv:2410.16336, doi: 10.48550/arXiv.2410.16336
Rauer, H., Aerts, C., Cabrera, J., et al. 2024, arXiv e-prints, arXiv:2406.05447, doi: 10.48550/arXiv.2406.05447
Richards, J. W., Starr, D. L., Butler, N. R., et al. 2011, ApJ, 733, 10, doi: 10.1088/0004-637X/733/1/10
Ricker, G. R., Winn, J. N., Vanderspek, R., et al. 2015, Journal of Astronomical Telescopes, Instruments, and Systems, 1, 014003, doi: 10.1117/1.JATIS.1.1.014003
Ripepi, V., Clementini, G., Molinaro, R., et al. 2023, A&A, 674, A17, doi: 10.1051/0004-6361/202243990
Rizhko, M., & Bloom, J. S. 2025, AJ, 170, 28, doi: 10.3847/1538-3881/adcbad
Roxburgh, H., Ridden-Harper, R., Moore, A., et al. 2025, arXiv e-prints, arXiv:2502.16905. https://arxiv.org/abs/2502.16905
Sarro, L. M., Debosscher, J., López, M., & Aerts, C. 2009, A&A, 494, 739, doi: 10.1051/0004-6361:200809918

Scargle, J. D. 1982, ApJ, 263, 835, doi: 10.1086/160554
Schuster, M., & Paliwal, K. 1997, IEEE Transactions on Signal Processing, 45, 2673, doi: 10.1109/78.650093
Shannon, C. E. 1948, Bell System Technical Journal, 27, 623, doi: 10.1002/j.1538-7305.1948.tb00917.x
Shen, J., Wu, W., & Xu, Q. 2024, arXiv e-prints, arXiv:2412.07997, doi: 10.48550/arXiv.2412.07997
Skarka, M., & Henzl, Z. 2024, A&A, 688, A25, doi: 10.1051/0004-6361/202450711
Skarka, M., Žák, J., Fedurco, M., et al. 2022, A&A, 666, A142, doi: 10.1051/0004-6361/202244037
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. 2014, J. Mach. Learn. Res., 15, 1929–1958
Stassun, K. G., Oelkers, R. J., Pepper, J., et al. 2018, AJ, 156, 102, doi: 10.3847/1538-3881/aad050
Tey, E., Moldovan, D., Kunimoto, M., et al. 2023, AJ, 165, 95, doi: 10.3847/1538-3881/acad85
TorchAudio. 2025, Pytorch. https://docs.pytorch.org/audio/main/_modules/torchaudio/models/conformer.html
Van Beeck, J., Bowman, D. M., Pedersen, M. G., et al. 2021, A&A, 655, A59, doi: 10.1051/0004-6361/202141572
Vaswani, A., Shazeer, N., Parmar, N., et al. 2017, arXiv e-prints, arXiv:1706.03762, doi: 10.48550/arXiv.1706.03762
Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: 10.1038/s41592-019-0686-2
Wang, Z., Yan, W., & Oates, T. 2016, arXiv e-prints, arXiv:1611.06455, doi: 10.48550/arXiv.1611.06455
Wen, Q., Zhou, T., Zhang, C., et al. 2022, arXiv e-prints, arXiv:2202.07125, doi: 10.48550/arXiv.2202.07125
Wu, Y., & He, K. 2018, in Proceedings of the European Conference on Computer Vision (ECCV). https://arxiv.org/abs/1803.08494
Zhang, J., Ye, L., & Lai, Y. 2023, Mathematics, 11, doi: 10.3390/math11091985
Zuo, S., Jiang, H., Li, Z., Zhao, T., & Zha, H. 2020, arXiv e-prints, arXiv:2002.09291, doi: 10.48550/arXiv.2002.09291