UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN COMPUTER ENGINEERING

"Neural Network Approaches for Classifying Variable Star Light Curves: Advancing Astrophysical Data Analysis"

Relatore: Prof. Loris Nanni

Laureando: Amedeo Carraro

Anno Accademico 2024–2025

Data di Laurea: 24/03/2025

Thank you to Professor Loris Nanni for his guidance and support.
To my family, who stressed me out just enough to keep me going.
To my friends, for never letting me feel alone.
To Martina, for believing in my intelligence.
To myself, for never doubting.
And to those who couldn't be here to see me reach this milestone.

"Basta un po' di ritmo."

Abstract

This thesis explores the application of two neural network architectures—Long Short-Term Memory (LSTM) and two-dimensional Convolutional Neural Networks (2D-CNN)—to the problem of classifying variable stars based on their light curves. Variable stars, such as Cepheids, BY Draconis, Delta Scuti, and Algol-type eclipsing binaries, provide fundamental information about stellar evolution and distance measurement, yet the manual analysis of their photometric variations is often slow and prone to errors.

Using a large dataset from the Zwicky Transient Facility (ZTF), LSTM and 2D-CNN models were developed, trained, and evaluated to automatically identify and distinguish different types of variable stars. The LSTM approach focuses on learning temporal dependencies in one-dimensional sequential data, while the 2D-CNN model leverages transformed representations of light curves to capture distinguishing features in a two-dimensional space. The results show that both models achieve high classification accuracy, significantly reducing the need for human intervention and accelerating the identification of new candidate variable stars. Moreover, this study highlights how deep learning methods can make the study of stellar variability more efficient and objective compared to traditional techniques.

In conclusion, the findings demonstrate the potential of data-driven approaches in astronomy: by combining astrophysical knowledge with advanced machine learning techniques, we obtain a powerful framework for analyzing large amounts of photometric data and deepening our understanding of the fundamental processes governing variable stars.

Abstract

Questa tesi esplora l'applicazione di due architetture di reti neurali — Long Short-Term Memory (LSTM) e Convolutional Neural Networks bidimensionali (2D-CNN) — al problema della classificazione delle stelle variabili a partire dalle loro curve di luce. Le stelle variabili, come le Cefeidi, le BY Draconis, le Delta Scuti e le binarie a eclisse di tipo Algol, forniscono informazioni fondamentali sull'evoluzione stellare e sulla misura delle distanze, ma l'analisi manuale delle loro variazioni fotometriche risulta spesso lenta e soggetta a errori.

Utilizzando un ampio dataset proveniente dallo Zwicky Transient Facility (ZTF), sono stati sviluppati, addestrati e valutati modelli basati su LSTM e 2D-CNN per identificare automaticamente e distinguere diverse tipologie di stelle variabili. L'approccio LSTM si concentra sull'apprendimento delle dipendenze temporali in dati sequenziali unidimensionali, mentre il modello 2D-CNN utilizza rappresentazioni trasformate delle curve di luce per catturare caratteristiche discriminanti in uno spazio bidimensionale. I risultati mostrano che entrambi i modelli raggiungono un'elevata accuratezza nella classificazione, riducendo significativamente il bisogno di intervento umano e accelerando l'identificazione di nuove candidate stelle variabili. Inoltre, questo studio evidenzia come i metodi di deep learning possano rendere più efficiente e oggettivo lo studio della variabilità stellare rispetto alle tecniche tradizionali.

In conclusione, i risultati dimostrano il potenziale degli approcci data-driven in ambito astronomico: combinando la conoscenza astrofisica con tecniche avanzate di machine learning, si ottiene un framework potente per analizzare grandi quantità di dati fotometrici e approfondire la comprensione dei processi fondamentali che regolano le stelle variabili.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Modern astronomical surveys are producing enormous volumes of time-series photometric data, driving a need for efficient automated methods to identify and classify variable stars. Variable stars—objects whose brightness changes over time—encode rich astrophysical information in their light curves, but traditional manual inspection or period-based classification is no longer feasible at scale [1]. In this context, deep learning techniques have emerged as powerful tools in the analysis of astrophysical data. Convolutional and recurrent neural networks can automatically learn complex patterns from data, and indeed 'many studies have shown that a deep learning framework such as CNN performs better than conventional machine learning algorithms' in various astronomy tasks[2].
Harnessing these advances for variable star classification is timely and essential, as the astronomy community enters the 'golden age' of time-domain surveys[1].
This thesis explores the application of deep learning models for classifying variable stars, focusing on two architectures: a two-dimensional Convolutional Neural Network (2D-CNN) and a Long Short-Term Memory (LSTM) recurrent network. The analysis is based on data from the Zwicky Transient Facility (ZTF), a contemporary sky survey representative of the new generation of high-cadence, wide-field observations[2].
The study is motivated by the increasing data volume and diversity of variable star-light curves, which demand scalable classification techniques. An investigation is conducted into the effectiveness of a 2D-CNN in classifying variable stars based on image-like representations of light curves, as well as the ability of an LSTM to extract temporal features from sequential photometric data. The relevance of this work lies in improving automated classification accuracy and efficiency, helping astronomers cope with data from current and upcoming surveys (such as the Vera Rubin Observatory's LSST) that will capture millions of variable sources.
In terms of structure, the thesis is organized as follows. The Introduction provides a historical and technical context, reviewing variable star classification efforts from early methods to modern deep learning approaches, as well as outlining the objectives of the study. The Preprocessing and Evaluation Metrics chapter then details the ZTF dataset, the data preprocessing steps, and introduces the evaluation metrics that will be used. The Methodology chapter presents the design of the CNN-LSTM and 2D-CNN base models, including a brief explanation of the un-

derlying LSTM and CNN architectures. Next, the Experiments and Results chapter presents the performance of each model and compares their strengths.

1.2 Background

Variable stars have long been studied for the insights they provide into stellar physics and galactic structure. Traditionally, classification of variable stars was performed by astronomers through careful observation: measuring periods and inspecting the shapes of light curves to assign stars to known classes (such as Cepheids, RR Lyrae, eclipsing binaries, etc.). Early catalogs and studies relied on phase-folded light curves and empirical criteria to distinguish types of variability. This manual and period-centric approach, while effective for small samples, became impractical as technology enabled surveys to monitor thousands of stars. The development of large-scale sky surveys in the late 20th and early 21st centuries (e.g. OGLE, WISE, Kepler, CRTS, TESS, Gaia) revolutionized time-domain astronomy by collecting abundant light curve data across large areas of the sky. These surveys provided unprecedented large sample sizes of variable stars, unlocking new science but also introducing the challenge of classifying vast datasets in an efficient and objective manner. The need for automation led researchers to devise algorithms that mimic the expert classification process.
A common early strategy for automated variable star classification was to extract descriptive features from each light curve and then apply traditional machine learning classifiers. These features encapsulate the variability characteristics—often split into periodic features (e.g. the star's dominant period, amplitudes and phases from Fourier decomposition of the light curve) and non-periodic or statistical features[3]. For example, in the study "Automated Classification of Variable Stars" by Debosscher et al. (2007)[4], 28 features were extracted from Fourier analysis (amplitudes, frequencies, etc.) and used for supervised classification. Similarly, Dubath et al. (2011), in their study "Random Forest Automated Supervised Classification of Hipparcos Periodic Variable Stars"[5], employed statistical moments as discriminative attributes for automated classification. Richards et al. (2011), in their study "On Machine-Learned Classification of Variable Stars with Sparse and Noisy Time-Series Data"[6], combined both periodic and descriptive features to classify large samples of variable stars, even exploring hierarchical classification schemes. To facilitate such approaches, tools like FATS (Feature Analysis for Time Series) and UPSILoN were developed, standardizing the computation of dozens of light curve features. UPSILoN, for instance, uses 16 engineered features and applies a Random Forest for classification. These feature-based methods demonstrated good accuracy on surveys like MACHO and OGLE, proving that machine learning could outperform purely manual classification for large datasets. However, they also revealed limitations. Notably, they assume each

light curve has a sufficient number of observations to reliably compute features (UPSILoN recommends >80 data points per light curve for robust performance). In practice, many survey light-curves are sparse, noisy, or have seasonal gaps due to observational constraints. Furthermore, features tuned to one survey's cadence or photometric bands may not generalize well to another survey with different sampling, introducing potential biases. These issues highlighted the need for more flexible classification approaches that do not rely heavily on perfect or dense light-curve sampling.

In recent years, the surge of deep learning has opened new avenues for variable star classification. Deep learning models can automatically learn salient patterns from raw data, reducing reliance on manual feature engineering. Indeed, the field has seen an 'upward trend' in developing classification models that ingest either the raw light curve or minimally processed data. The motivation is to let neural networks discover the features optimally, which is especially useful for irregular or complex light curves that elude simple parametric descriptions. For instance, Mahabal et al. (2017), in their study "Deep-Learnt Classification of Light Curves"[7], pioneered a two-dimensional image-based approach. They transformed each light curve into a dm–dt image, a binned 2D histogram representing the differences in magnitude and time between all observational pairs, and used a Convolutional Neural Network (CNN) to classify variable stars based on these representations. This allowed the CNN to learn variability patterns in both time and magnitude domains simultaneously. Naul et al. (2018), in the work "A Recurrent Neural Network for Classification of Unevenly Sampled Variable Stars"[8], took a different route by training an RNN autoencoder on unlabeled light curves to learn a compressed representation; the latent features from this autoencoder (especially when using period-folded inputs) were then fed to a classifier, yielding an effective semi-unsupervised solution. Other studies have explored hybrid architectures: The methodology introduced by Aguirre et al. (2019) in "Deep Multi-Survey Classification of Variable Stars"[9] computed sequential difference vectors for time and magnitude and organized them as a 2-channel matrix input to a 1D CNN, effectively capturing the light curve's incremental changes. Similarly, recurrent-convolutional models (RCNNs) have been applied to sequences of images or photometric data, achieving high recall (~94%) on transient survey data, while pure RNN models (e.g. LSTM or GRU networks) have reached accuracies of ~95% on classifying main variability types. These successes underscore the potential of deep learning: numerous studies report that CNNs and RNNs can equal or surpass traditional classifiers in this domain. Deep models can handle light curves of varying length and learn both short-term and long-term dependencies in the data, an advantage when dealing with the diverse time scales of stellar variability. On the other hand, deep learning methods typically require large training sets and can be computationally intensive—fortunately, contemporary surveys often provide the requisite data volume, and computing resources continue to grow.

Despite significant progress, several challenges persist in classifying variable stars, which this study must keep in mind. Class imbalance is a common issue: sky surveys tend to find huge numbers of common variable types (e.g. RR Lyrae, eclipsing binaries) but far fewer examples of rare classes, leading to highly skewed class distributions[1]. Imbalanced datasets can bias a model to favor majority classes, so special care (such as resampling or cost-sensitive training) is needed to properly recognize minority classes. Another challenge is the presence of gaps and irregular sampling in light curves. Ground-based surveys like ZTF have daytime gaps and weather interruptions, yielding uneven. Irregular cadence and missing data points can confuse classifiers or period-finding algorithms, making methods that can handle sequence variability and missingness (like LSTMs or interpolation techniques) especially valuable. Additionally, real survey data often contain outliers and noise that must be mitigated during preprocessing to avoid degrading model performance. A further consideration is the interpretability of machine learning models. Traditional feature-based classifiers allowed some insight into which attributes (period, amplitude, etc.) drove the classification, whereas deep neural networks operate as "black boxes" with many parameters, making it harder to trace how a classification decision was reached. This lack of interpretability can be problematic in astrophysics, where understanding the reasoning behind a classification (to possibly relate it to physics) is desirable. Efforts are underway in the community to apply explainable AI techniques or calibrate model confidence [10], but interpretability remains an open challenge. By acknowledging these issues—such as data imbalance, incomplete light curves, and limited model transparency—the methodology is designed to address them as effectively as possible through the use of appropriate evaluation metrics and careful inspection of model outputs, as discussed in the following sections.

1.3 Objectives of the study

The primary objective of this thesis is to develop and evaluate deep learning models for automated classification of variable stars using the ZTF light curve dataset. In particular, the focus is placed on two complementary model architectures:

- a recurrent LSTM network that processes one-dimensional time-series data
- a two-dimensional CNN that operates on image-like representations of light curves

By pursuing these two approaches, the objective is to assess how sequence-based and image-based deep learning methods compare in the classification of a diverse set of variable stars. The choice of the ZTF dataset is motivated by its scale and relevance: ZTF is a modern time-domain survey providing nearly one million of photometric light curves in multiple bands [11], serving as an excellent proxy for the data volumes expected from upcoming projects like LSST. ZTF's public catalog contains a variety of known variable stars with labels (types) that can be used for supervised learning, and its high-cadence observations align with the goal of testing models on realistic, large astronomical datasets. Developing the models on ZTF data ensures that the approach is grounded in practical data challenges and that the resulting findings are directly applicable to current survey pipelines.

This work addresses several research questions: How effectively can an LSTM model learn from raw, irregularly sampled light curves to classify variable stars compared to a 2D-CNN trained on transformed light curve images? Which model architecture demonstrates greater robustness to challenges such as missing observations or noise, and do the resulting classification errors reveal different characteristics of the data? What levels of classification performance—measured in terms of accuracy, precision, and recall—can be achieved by these deep learning models on the ZTF dataset, and do they outperform traditional feature-based classifiers documented in the literature? Furthermore, the study explores whether integrating insights from both models can enhance overall classification performance or support the identification of rare variable types. Through these questions, the thesis evaluates the hypothesis that deep neural networks can substantially improve the automation and accuracy of variable star classification, while also considering the trade-offs associated with each architectural approach.

1.4 Limitations

Despite the promising nature of the developed neural network architectures, several limitations related to their design, training procedures, and the experimental setup merit consideration.

Firstly, the neural networks were trained and tested exclusively on the Zwicky Transient Facility (ZTF) Catalog 2 dataset. Although this dataset is extensive and representative of modern astronomical surveys, reliance on a single source limits the generalizability of the models. Each astronomical survey has unique observational characteristics, sampling cadences, photometric noise profiles, and biases. Therefore, a model trained on the ZTF dataset might not achieve comparable performance when applied to data from other surveys, such as Gaia, ASAS-SN, or LSST, without additional training or adjustments.

Secondly, the hardware used for training—a 12th Gen Intel(R) Core(TM) i7-12700H CPU running at 2.30 GHz, 16 GB RAM, and an NVIDIA GeForce RTX 3050 Ti laptop GPU—imposed substantial computational constraints. These limitations significantly influenced the depth of hyperparameter optimization and experimentation. Training computationally demanding models, particularly the 2D-CNN architecture, required extensive resources and considerable time, limiting experimentation with more complex or deeper neural network architectures. Additionally, memory constraints occasionally necessitated reductions in batch size, potentially influencing model convergence.

Thirdly, class imbalance in the dataset posed a significant challenge, even after balancing procedures. While random undersampling partially alleviated this issue, it inevitably resulted in the loss of informative examples from majority classes. Furthermore, rare classes remained underrepresented, potentially limiting the model's ability to generalize effectively to these categories.

Fourthly, the input data preprocessing, specifically the use of harmonic modeling and period-based reconstruction to create uniform sequences for LSTM models, inherently simplifies the complex nature of real astronomical time series. Such simplifications may not fully capture the variability present in raw, irregularly sampled observational data. Advanced preprocessing techniques or representation-learning methods more suitable for irregular and noisy time-series data might provide improvements.

Lastly, interpretability remains a critical limitation. Both CNN and LSTM architectures function largely as "black-box" models, making it challenging to discern precisely which features or patterns drive specific classifications. This lack of interpretability complicates the verification of model predictions and reduces trust within the astronomical community, where explanatory power often holds considerable importance.

2 Preprocessing and Evaluation Metrics

2.1 Data

The Zwicky Transient Facility (ZTF) dataset is a large astronomical survey program in the time domain, which repeatedly monitors nearly a million celestial objects to identify brightness variations. This dataset collects a series of characteristic parameters of the light curves of variable stars, along with the variable class each star belongs to. In other words, for each identified variable star, its key photometric properties (such as period, magnitudes, amplitude, etc.) and its variable type (e.g., RR Lyrae, Cepheid, Eclipsing, etc.), known from previous classifications, are reported. This information serves as the knowledge base for training and evaluating neural networks for automatic classification. Only the key columns utilized as model inputs or targets are discussed here, while unnecessary fields (e.g., coordinates or IDs) are omitted for clarity.
One of the most important features—if not the most important—is the period (Per), which indicates the variability period of the star, expressed in days. The period represents the time it takes for the star to complete a full cycle of brightness variation. This is a crucial parameter, as many classes of variable stars are defined by specific period ranges. For example, classical Cepheids typically have periods ranging from a few days to several tens of days, RR Lyrae stars around half a day, eclipsing variable stars can have periods from a few hours to many days, while Miras (pulsating giant stars) can have periods of several hundred days. Knowing the period therefore allows for an initial rough distinction between different categories of variable stars.
In addition to the period, the dataset provides magnitude measurements in two photometric bands. One of these is rmag, which represents the average (or reference) magnitude of the star in the red (r) filter of the ZTF system. Magnitude is an inverse logarithmic scale of brightness: higher values indicate fainter stars. In the context of variable stars, the average magnitude offers insight into the object's brightness and, indirectly, its distance or intrinsic nature. For example, RR Lyrae and Cepheids tend to be intrinsically bright stars, while other types, such as eclipsing binaries, may consist of fainter systems. In this dataset, rmag serves as a descriptive parameter and may also contribute to further classification analysis.
Another fundamental parameter is the amplitude of the brightness variation, generally provided as Amp_g and Amp_r, which represent the peak-to-peak amplitude of the magnitude oscillation in the green and red bands, green corresponding to shorter wavelengths (~ 480 nm) and red to longer wavelengths (~ 620 nm). The amplitude indicates how many magnitudes the

star's brightness changes between the minimum and maximum of its cycle. This parameter is also highly diagnostic: some classes of variable stars exhibit very large variations (for example, classical Cepheids and Miras can vary by several magnitudes, making their changes easily noticeable to the naked eye), while others show more modest variations.
In addition to the period, mean magnitudes, and amplitudes, ZTF includes parameters derived from harmonic fits of the light curves. Specifically, it reports R21 and phi21, which describe the shape of the light curve through Fourier series decomposition. These parameters result from fitting the light curve with a periodic function composed of a fundamental frequency and its harmonics (typically twice the fundamental frequency for the first harmonic, three times for the second, and so on). R21 is defined as the amplitude ratio between the second harmonic and the fundamental (formally, R21 = A2/A1). It is a dimensionless number that quantifies how significant the second harmonic peak is compared to the main one. phi21 (also written as ϕ_21) is the phase difference between the second harmonic and the fundamental, often expressed as ϕ_2 - 2ϕ_1, where ϕ_1 is the phase of the fundamental and ϕ_2 is that of the second harmonic. In practice, R21 and phi21 together characterize the asymmetry of the light curve. For example, pulsating stars like Cepheids and RR Lyrae typically exhibit non-sinusoidal light curves with sharp peaks and slower descents—features that result in a non-zero R21 and a distinctive phi21 phase offset. Finally, the "Type" feature identifies the ten types of stars present in the dataset, namely:

- 'BYDra' (BY Draconis),
- 'CEP' (Cepheid Variable + Type II Cepheid),
- 'DSCT' (Delta Scuti),
- 'EA' (Eclipsing Binary Algol),
- 'EW' (Eclipsing W Ursae Majoris),
- 'Mira' (Mira Variable),
- 'RR' (RRab Lyrae),
- 'RRc' (RRc subtype),
- 'RSCVN' (RS Canum Venaticorum),
- 'SR' (Semi-regular Variable).

2.1.1 Data Cleaning

Before feeding the data into these models, a thorough cleaning and preprocessing procedure is carried out to enhance the quality and reliability of the dataset. The raw dataset (as obtained from the survey catalog) may include entries with unreliable period estimates, insufficient observations, or inconsistent labeling. A series of filtering steps with predefined thresholds is applied to refine the dataset. These steps remove poor-quality data and standardize the inputs, which in turn enhances model performance by focusing learning on true signals rather than noise. The main data cleaning procedures are as follows:

- Minimum Detections Filter: Each star is required to have a sufficient number of observations in both photometric bands. In practice, any star with fewer than 50 data points in either g or r band is excluded. This threshold ensures the retention of only those light curves that contain a sufficient number of measurements to reliably determine variability parameters. A light curve with very sparse data (e.g., 10–20 points) might lead to an incorrect period or amplitude estimation; by filtering out such cases, the overall fidelity of the input features is improved. This step reduces the chance of the models being confused by artifacts of undersampling.

- Period Confidence (False Alarm Probability):Stars with high period uncertainty are filtered out using the False Alarm Probability (FAP) associated with period detection. Specifically, only entries with log(FAP) < –3 in both the g and r bands—corresponding to a false alarm probability below 0.001—are retained. This criterion means that the likelihood of the period being spurious is at most 0.1%. By discarding stars with higher FAP, this ensures that nearly all periods provided to the models are real and statistically significant. This dramatically improves data quality because variable star classification heavily relies on an accurate period; a wrong period could mischaracterize the light curve shape and mislead the model.

- Goodness-of-Fit (R^2) Threshold: Each light curve in the catalog comes with a goodness-of-fit metric (R^2) for the period model in each band (indicating how well a periodic model explains the observed variance). A minimum quality threshold of R^2 > 0.3 is enforced for both g and r band fits. Light curves with R^2 below this threshold are likely not well-described by a stable periodic signal – perhaps due to irregular variability or high noise. Removing these cases means the remaining stars exhibit reasonably periodic behavior that the models can learn. In essence, this step filters out objects whose variability is not clearly periodic (or not captured by the simple model), thereby honing the training set to true periodic variables.

- Type Label Corrections: Class labels are normalized to prevent duplication or ambiguity. One example is consolidating "CEPII" into "CEP"– if the catalog distinguished Type II Cepheids separately, they are merged into the general Cepheid class for simplicity. Similarly, any spelling variations or sub-class distinctions that are not needed for this classification scope are unified. This ensures that the label "CEP" consistently refers to all Cepheid variables in the training data. Such corrections prevent the model from being confused by what are essentially the same physical class labeled differently. It also reduces the number of output categories if some were too fine-grained, focusing the classification task on the major types of interest.

- Removal of Anomalous Entries: Any remaining entries with clearly invalid or extreme values in the critical columns are removed. For instance, if a star has a period of 0 or an amplitude of 0 (which might indicate a bad fit or placeholder value), that row is removed. Likewise, any non-physical values (e.g., negative period) or missing data are handled at this stage. The removal of such anomalies prevents the inclusion of erroneous data that could distort model training or introduce outliers unrepresentative of genuine astrophysical behavior.

- Class Balancing: Following the application of the above filters, the distribution of star types within the dataset is examined. Often, some classes (such as Eclipsing Binaries or certain pulsating stars) are far more numerous than others. To prevent the models from being biased toward the majority class, the dataset is balanced by limiting the number of examples per class. In this case, each class was capped at a maximum of 1,000 instances through random under-sampling when the number of stars in a class exceeded this threshold. This balances the training data across classes, so that common types do not overwhelm rarer ones during learning. A more balanced dataset helps the models because it forces them to learn distinguishing features of all classes, rather than simply predicting the dominant class most of the time. (It is noted that ideally, very small classes would be oversampled or augmented; however, in this dataset, all selected classes retained a reasonably large population after filtering.) Finally, a 5-fold cross-validation is performed by randomly splitting the stars into five groups, enabling multiple testing iterations.

After these cleaning steps, the resulting dataset is of high quality: each star has a well-defined period with strong statistical significance, a sufficiently sampled light curve, and a reliable class label. The features (period, amplitudes, etc.) are thus trustworthy descriptors of the star's variability. By enhancing the signal-to-noise ratio and ensuring consistency—through label cleaning and class balancing—a solid foundation is established for training the LSTM and CNN models. Empirically, this preprocessing leads to better model convergence and accuracy, as the algorithms can focus on real variability patterns rather than contending with noisy or mislabeled examples. In summary, data cleaning greatly enhances model performance by filtering out confounding factors and presenting the learning algorithms with data that truly represents the distinct characteristics of each variable star class.

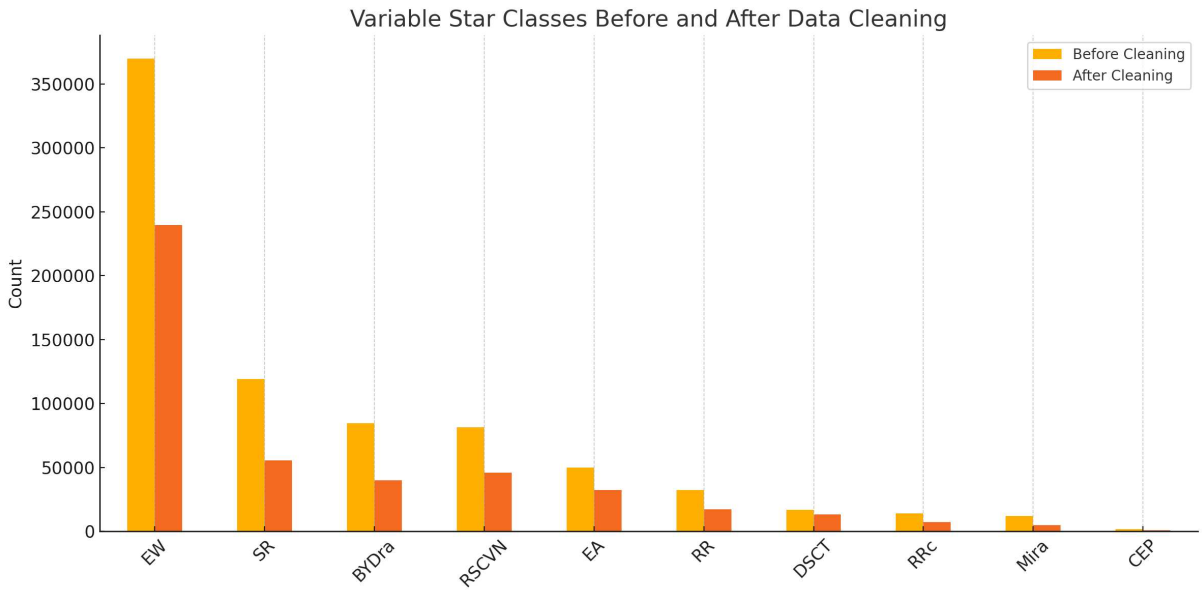

Figure 1: Number of stars before and after cleaning.

2.2 Image Generation

The neural networks developed in this work will rely on two distinct types of images as input. First, light curves, which capture flux variations of an object over time, will be transformed into a 2D representation suitable for network processing. Second, DMDT images—derived from difference distributions across multiple time intervals—will provide an additional perspective on temporal changes. By generating and combining these two complementary image forms, the models are provided with informative visual representations designed to capture both the temporal evolution and the structural characteristics of the stellar variability patterns.

2.2.1 Light curves Mappings

LSTM (Long Short-Term Memory) networks are designed to handle sequential or temporal data by capturing dependencies over time. In the context of variable stars, the natural input for an LSTM would be the light curve—that is, the sequence of a star's brightness over time. However, there are several complexities: astronomical light curves are often irregularly sampled (i.e., observations are not equally spaced in time), vary in length from star to star (different numbers of data points), and may contain gaps. Standard LSTMs generally require input sequences to have a uniform length and, ideally, regular sampling.

To address these challenges, rather than inputting raw observational data into the LSTM, extracted harmonic parameters (period, amplitude, R21, phi21) are used to reconstruct a regularized version of the light curve for each star. Specifically, the following procedure is applied

to each star: using the period Per and the Fourier fit parameters (amplitude in r, R21, and phi21), a modeled light curve is constructed to approximate the star's magnitude variation over a complete cycle, ranging from phase 0 to phase 1. (i.e., one full period). The parameter rmag is interpreted as a reference magnitude level (for example, the mean or peak brightness, depending on how it is defined in the catalog; typically, in this type of fit, it corresponds to the mean magnitude).
A modeled function of the following form is considered:

$$m(\phi) = M + A_1 \cos(2\pi\phi) + B_1 \sin(2\pi\phi) + A_2 \cos(4\pi\phi) + B_2 \sin(4\pi\phi)$$

where ϕ is the phase (ranging from 0 to 1 over the period), M is the offset term (set equal to rmag), and the coefficients A_1, B_1, A_2, B_2 are determined based on Amp_r, R21, and phi21. Specifically, if Amp_r is the peak-to-peak amplitude in the r band, the fundamental component (first harmonic) has an amplitude approximately equal to half of Amp_r (since in a pure sine wave, the peak-to-peak range is twice the amplitude of the first harmonic). Therefore, the values are set to A_1 = 0.5 × Amp_r, and A_2 = A_1 × R21 (applying the amplitude ratio to compute the second harmonic's amplitude).

The initial phase angles for the cosine and sine terms are chosen so that the fundamental has zero phase (i.e., a maximum at the start of the cycle, ϕ_1 = 0), while the second harmonic is phase-shifted by phi21 relative to twice the fundamental frequency (ϕ_2 = phi21, in radians). From these, the coefficients are calculated as:

$$A_i = A_i \cdot \cos(\phi_i) \quad \text{and} \quad B_i = A_i \cdot \sin(\phi_i) \quad i = 1, 2.$$

The result is that m(phi_i) represents a synthetic light curve that incorporates both the basic brightness variation and a degree of harmonic structure to account for non-sinusoidal features.

Once the parametric model is defined, a sequence of light curve points is generated by dividing the phase interval [0, 1] into a fixed number of steps (in this case, 360) and computing m(ϕ) for each point. This yields a magnitude profile as a function of phase, with 0–359 values corresponding to one full cycle of the star.

Since phase is a normalized variable (0 at the start of the cycle, 1 at the end, regardless of the actual duration in days), all stars have sequences of equal length and can be directly compared in terms of their behavior over one period. In other words, the light curves are transformed into standardized waveforms: for example, an RR Lyrae of type ab will show a rapid brightness increase (i.e., drop in magnitude) and a more gradual decline, while an EA-type eclipsing binary will display two dips corresponding to the primary and secondary eclipses (with the primary being deeper), spaced throughout the cycle.Each star is thus represented by a fixed-length time series vector of magnitudes.

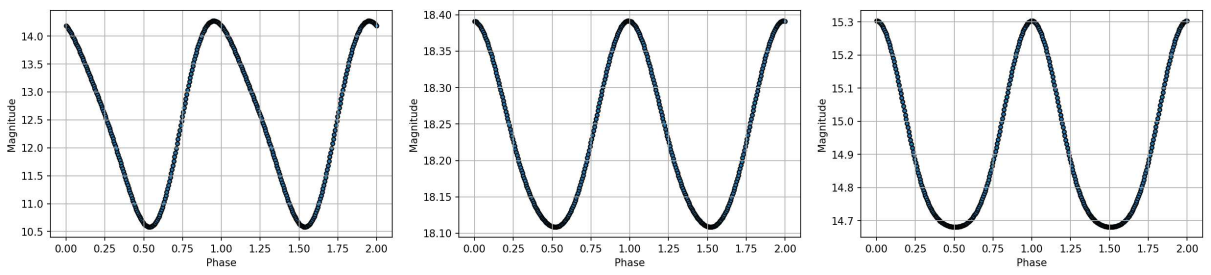

Figure 2: Three examples Light curves images: in order,'Mira' (Mira Variable), 'EW' (Eclipsing W Ursae Majoris), 'CEP' (Cepheid Variable), the same stars as before.

2.2.2 DMDT Mapping

Another strategy involves converting the variability features of each star into a two-dimensional image, allowing the use of 2D convolutional neural networks. The idea behind this representation is inspired by recent approaches that aim to visualize the light curve in a specific space, often referred to as the $\Delta m - \Delta t$ space, where Δm (delta magnitude) and Δt (delta time) represent the magnitude variation and the corresponding time difference between observations, respectively. In the literature[7], this technique has been proposed as a compact way to capture the variability behavior: a single image condenses information about how much a star's brightness changes and how quickly those changes occur.
The expectation is that different classes of variable stars will produce distinctive patterns in this 2D space, which can be recognized by a visual classifier. Ideally, to construct the $\Delta m - \Delta t$ representation of a light curve, one considers all possible pairs of observations and computes the difference in magnitude between them ($\Delta m = m_j - m_i$, or its absolute value to focus on the size of the change) and the absolute time difference ($\Delta t = |t_j - t_i|$).
For instance, if a star has a well-defined period P, there will be many point pairs for which Δt is approximately P (or its multiples), and the corresponding Δm could be close to the maximum amplitude (e.g., when comparing maximum and minimum brightness). For periodic variables, one expects clusters of (Δt, Δm) points around the fundamental period with high Δt (amplitude), and possibly at multiples of the period with slightly lower Δm values (if not all cycles have the same range). For an eclipsing star with unequal minima, clusters may appear at Δt $\approx$ P and $\Delta t \approx$ P/2 . An irregular pulsating star, by contrast, might display more scattered Δm values even for small Δt, indicating unpredictable changes on short timescales.
These (Δt, Δm) pairs are then discretized into a 2D histogram, dividing the $\Delta m - \Delta t$ space into bins and counting how many pairs fall into each bin. The result can be interpreted as a grayscale image, where the intensity of each pixel corresponds to the number (or frequency) of occurrences of a given combination of time and magnitude variation. This type of image reveals both the typical amplitude scales (on the Δm axis) and typical temporal scales (on the Δt axis) of the star's variability.
In other words, it's a map of how the star changes: a bright spot in the upper-left corner of the image, for instance, may indicate many occurrences of small magnitude changes over very

short timescales (typical of noisy or rapidly micro-variable stars), whereas bright areas toward the upper right (large Δm and large Δt) suggest that the star undergoes dramatic brightness changes over long timescales (as in long-period variables). The shapes and relative intensities of these regions in the image act as a visual signature of the variable star's class.
In this case, a simplified yet effective $\Delta m - \Delta t$ representation is constructed, primarily based on period and amplitude information. Instead, a comparative approach across stars within the same class is adopted to generate the $\Delta m - \Delta t$ patterns. Specifically, for each star within a given class, differences are calculated relative to other stars of the same class—namely, the difference in g-band amplitude, r-band amplitude, and period. Formally, for two stars i and j, the following values are computed:

- $\Delta m_g = \left|\mathrm{Amp}_{g,i} - \mathrm{Amp}_{g,j}\right|$
- $\Delta m_r = \left|\mathrm{Amp}_{r,i} - \mathrm{Amp}_{r,j}\right|$
- $\Delta t = |\mathrm{Per}_i - \mathrm{Per}_j|$

The choice of using amplitude differences between stars stems from the desire to simulate the range of possible magnitude variations within that class, while the difference in period reflects the range of characteristic timescales. Although this method differs from calculating intra-curve differences, it still creates a set of (Δt, Δm) points representing how much two similar variables can differ—emphasizing the typical amplitude and period ranges for that class.
Once the list of (Δt, Δm) pairs for a star is obtained (from comparisons with other stars in the same class), a 2D histogram is constructed. The Δt axis is divided into predefined bins ranging from the smallest to the largest period differences in the dataset, and similarly for Δm—from very small amplitude differences to several magnitudes.
For instance, the Δm bins in the code were defined as: [0, 0.1, 0.2, 0.3, 0.5, 1, 1.5, 2, 2.5, 3, 5, 8] (in magnitudes), and the Δt bins range from time differences of minutes/hours up to decades, starting from fractions of a day (e.g., 1/145, 1/25 days – corresponding to a few minutes or about an hour) up to values like 4000 days (more than a decade), divided logarithmically/non-uniformly to give higher resolution to small time differences, which are crucial for distinguishing fast variables.
Each computed pair falls into one of these time and amplitude intervals, incrementing the count for that cell. The result is a numeric grid (matrix) representing the density of (Δt, Δm) points for that star.

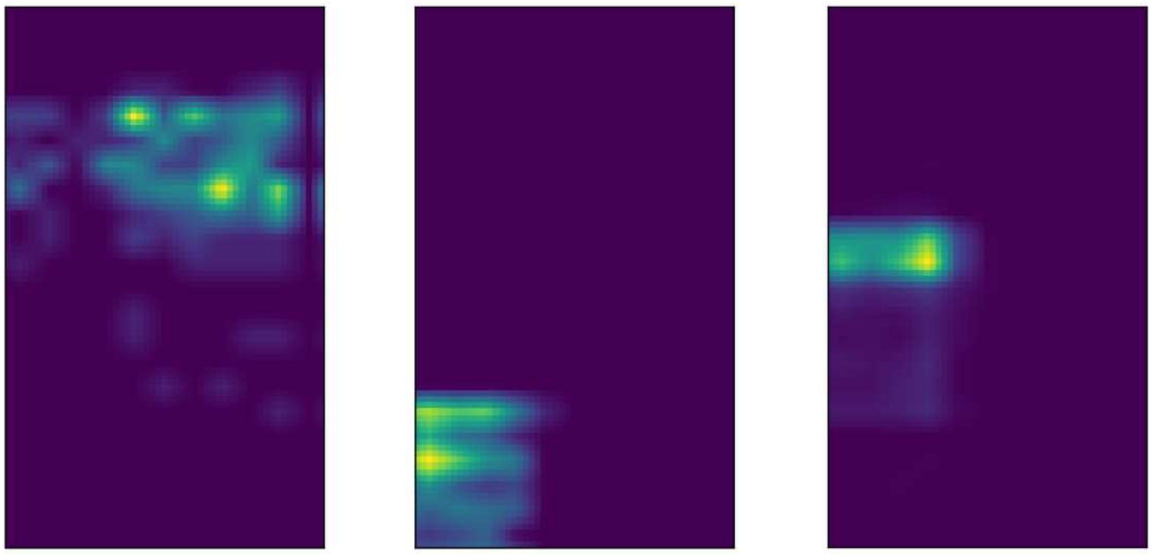

Figure 3: Three examples of dm–dt images: in order,'Mira' (Mira Variable), 'EW' (Eclipsing W Ursae Majoris), 'CEP' (Cepheid Variable).

2.3 Evaluation Metrics

In machine learning classification, various metrics are used to evaluate model performance beyond just overall error. Key metrics include Accuracy, Precision, Recall, F1-Score to summarize outcomes[23] and perhaps the confusion matrix to identify which classes are commonly confused . Each metric provides different insights into the model's strengths and weaknesses[24]. Below, each metric is defined, its mathematical formula is provided, and its usefulness is discussed.

2.3.1 Accuracy

Accuracy is the proportion of total predictions that the model got correct[25]. It is calculated as the number of True Positives (TP) plus True Negatives (TN), divided by the total number of predictions (TP + TN + False Positives (FP) + False Negatives (FN)). Formally:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

This metric is very intuitive and widely used—for example, 90% accuracy means 90 out of 100 predictions were correct. Accuracy is reliable when the dataset is balanced and each class is equally important[26]. In such cases (e.g., when positive and negative classes are of similar frequency), accuracy gives a quick snapshot of performance. However, accuracy can be misleading in imbalanced datasets. If one class vastly outnumbers the other, a model can achieve high accuracy by simply predicting the majority class every time, yet it performs poorly on the minority class.

2.3.2 Precision

Precision (also called Positive Predictive Value) measures how many of the instances that the model predicted as positive are actually positive[24]. It focuses on the accuracy of positive predictions, thus emphasizing the cost of false alarms. The formula for precision is:

$$\text{Precision} = \frac{TP}{TP + FP}$$

A higher precision means that fewer false positives being made, that is, when the model flags a positive, it is likely correct. This metric is crucial in scenarios where false positives are particularly undesirable or costly. For instance, in medical diagnosis, a test with high precision ensures that patients who are told they tested positive truly have the condition, avoiding unnecessary anxiety or treatment for those who do not.

2.3.3 Recall (Sensitivity)

Recall, also known as Sensitivity or the True Positive Rate, measures how many of the actual positive instances the model correctly identified. It is defined as the number of true positives divided by the total actual positives (TP + FN):

$$\text{Recall} = \frac{TP}{TP + FN}$$

High recall means the model finds most of the positive cases, minimizing false negatives. This metric is critical when missing a positive instance has a high cost. For example, in disease detection (medical diagnosis), failing to diagnose a patient who actually has the disease (a false negative) can be life-threatening.

2.3.4 F1-Score

The F1-Score is a single metric that combines precision and recall into their harmonic mean[25]. It provides a balanced evaluation when both false positives and false negatives need to be taken into account. The F1-score is given by:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

This formulation means the F1-score will only be high if both precision and recall are high; it penalizes extreme trade-offs between the two. For example, if precision is 1.0 but recall is 0 (or vice versa), the F1 will be 0, reflecting the poor overall performance despite one strong metric. Because it balances precision and recall, the F1-score is useful in contexts where a compromise between avoiding false positives and false negatives is sought. It is especially important in imbalanced datasets, where a high accuracy might be achieved by ignoring the minority class. In such cases, precision and recall (and thus F1) give a more meaningful picture of performance. A model with a high F1-score has a good balance, meaning it is catching a large portion of the positive instances while also keeping false positives low. Therefore, the F1-score is often used as a summary metric when one wants to ensure the model performs well on the positive class without being misled by class imbalance or by focusing on only one aspect of errors.

2.3.5 Confusion Matrix

The confusion matrix can be interpreted as a structured representation of classification performance, offering detailed insight into the distribution of true versus predicted labels across categories—often characterized by varying degrees of difficulty (e.g., easy, medium, hard). Specifically, the primary diagonal cells illustrate the number of accurate predictions by the model, whereas the off-diagonal cells highlight misclassifications. For example, it makes clear cases such as when a label classified as "easy" is actually "hard," pinpointing exactly where the model struggles (see Appendix). The confusion matrix frequently employs a color gradient as an intuitive indicator of prediction frequency, with darker hues indicating a higher occurrence of certain prediction outcomes. This visualization technique provides critical insights into the predictive model's generalization capabilities, going beyond mere aggregate performance metrics, and helps identify specific strengths and weaknesses in classification across distinct categories or difficulty levels[31][32].

3 Methodology

3.1 Basic Neural Network

This section provides a brief introduction to the functioning of basic LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) models. These neural networks are commonly used to process sequential and structured data, respectively. By combining them, it's possible to leverage the strengths of both architectures. Following this brief overview, the focus shifts to the custom model developed specifically for the project.

3.1.1 Long-Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks[12][13] are a specialized form of recurrent neural network (RNN) designed to address the vanishing gradient problem and capture long-term dependencies in sequence data. LSTM cells incorporate three gates—input, forget, and output—and maintain two hidden "states" across time steps: the hidden state H_t and the cell state C_t. The cell state is governed by a "forget gate" that selectively discards irrelevant information from the previous cell state, and an "input gate" that adds relevant new information.

Given an input X_t and the previous hidden state H_{t-1}, each gate is computed via:

$$F_t = \sigma\Big(U_f\,X_t + W_f\,H_{t-1} + b_f\Big),$$

$$I_t = \sigma\Big(U_i\,X_t + W_i\,H_{t-1} + b_i\Big),$$

$$O_t = \sigma\Big(U_o\,X_t + W_o\,H_{t-1} + b_o\Big).$$

A candidate cell state $\tilde{C}_t$ is generated via a $\tanh$ layer:

$$\tilde{C}_t = \tanh\Big(U_c\,X_t + W_c\,H_{t-1} + b_c\Big).$$

Next, the cell state C_t is updated by combining the old cell state C_{t-1}, scaled by the forget gate,

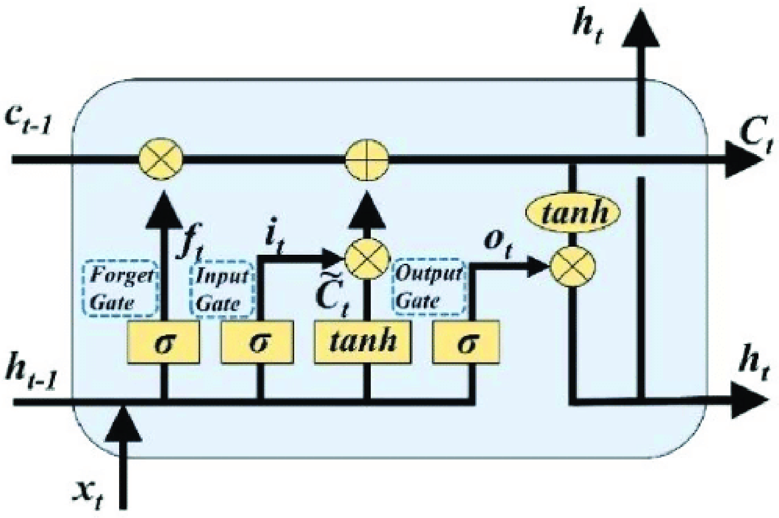

Figure 4: Architecture of LSTM.

and the candidate cell state, scaled by the input gate:

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t,$$

where $\odot$ denotes elementwise multiplication. Finally, the hidden state H_t is computed as:

$$H_t = O_t \odot \tanh(C_t).$$

Thanks to the gating mechanisms, LSTMs can selectively retain pertinent information over extended sequences, which makes them highly effective in tasks such as language modeling, time-series forecasting, and sequence labeling. Peephole connections[14] and other modifications[15] have also been proposed to further refine LSTM performance. Today, LSTM-based architectures remain foundational in diverse applications, especially where data exhibit long-range temporal dependencies.

3.1.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs)[16] have emerged as a cornerstone of modern image analysis and have demonstrated exceptional capabilities across a wide range of computer vision applications. Their impact is particularly prominent in medical imaging, where CNN-based methods have achieved remarkable results in tasks such as lesion detection and organ segmentation, often surpassing traditional techniques that rely on hand-crafted features. Building upon the principles of deep learning, CNNs automatically extract and refine hierarchical representations from large datasets, which is critical for accurately capturing complex visual patterns[17].

At the core of a CNN is the convolutional layer, which performs a two-dimensional (2D) convolution over input data—be it a raw image or an intermediate feature map. This operation can be mathematically described as:

$$(F * X)[i, j] = \sum_{a=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{b=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} F[a, b] \cdot X[i - a,\, j - b],$$

where X denotes the input feature map and F is a learnable kernel of size $k \times k$. Convolution filters like F effectively scan the spatial dimensions of X, computing localized, weighted sums that emphasize edges, textures, or other discernible features. After this operation, each filter's output is often followed by a learnable bias term and a non-linear activation function such as the Rectified Linear Unit (ReLU). This non-linearity is key to modeling complex relationships in images.

CNNs commonly stack multiple convolutional layers, forming a deep architecture that sequentially refines feature representations. In the initial layers, kernels typically pick up simple geometric elements like edges and corners, whereas deeper layers integrate these basic patterns into more abstract concepts such as shapes, object parts, or entire structures. This progressive abstraction allows CNNs to handle intricate visual details and significantly reduces the need for manual feature engineering. In the medical domain, where subtle differences in pixel intensity or texture may indicate critical pathology, CNNs have proven particularly effective.

Beyond convolution, modern CNN architectures often incorporate pooling layers to reduce the spatial dimensions of feature maps, batch normalization to stabilize and speed up training, dropout to mitigate overfitting, and fully connected layers to fuse features for final classification or regression tasks. Together, these components form a robust framework that exploits the hierarchical nature of image data, enabling CNNs to tackle diverse challenges in segmentation, detection, and recognition with high accuracy and efficiency.

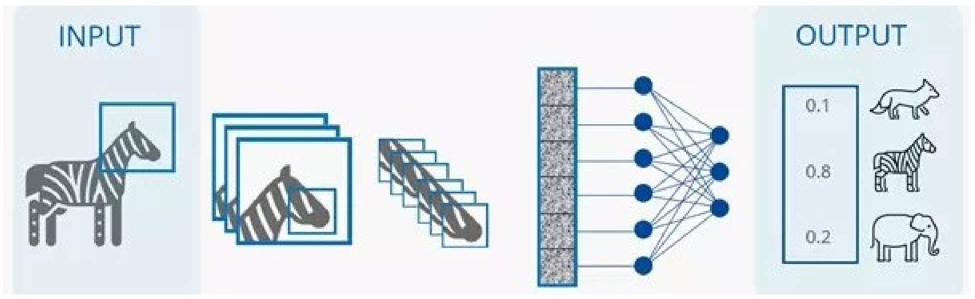

Figure 5: Architecture of CNN.

3.1.3 Pooling Layers

Pooling layers are fundamental components of CNN architectures, designed to reduce the spatial dimensions of feature maps, thereby lowering computational complexity and decreasing memory requirements. According to LeCun et al. (2015) [17], pooling contributes significantly to the CNNs' ability to learn hierarchical feature representations. Among the various pooling methods, max pooling is the most widely adopted. Max pooling selects the highest activation within a defined local region (pooling window), effectively retaining only the most salient features and discarding redundant spatial information.

Mathematically, the max pooling operation is defined as:

$$z(j) = \max_{u \in \Omega(j)} \{a_u\},$$

where $\Omega(j)$ represents the set of spatial positions within the pooling window centered at index j, and a_u denotes the activation at position u. This procedure not only reduces the computational complexity of subsequent layers but also introduces translation invariance, enhancing the robustness of the network to minor positional variations in the input data. Moreover, pooling layers contribute to regularization by reducing overfitting, which is particularly beneficial in applications with limited training data [18][19].

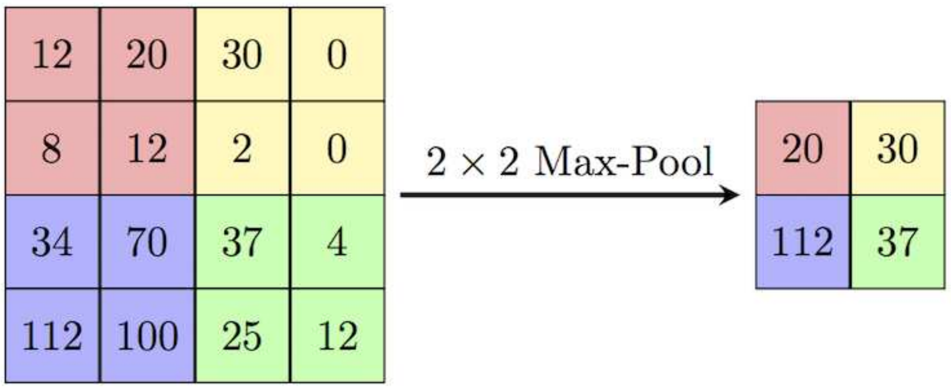

Figure 6: Max Pooling process.

3.1.4 Batch Normalization

Batch Normalization is a technique introduced by Sergey Ioffe and Christian Szegedy (2015)[19] to stabilize and accelerate the training of deep neural networks. During training, the distribution of each layer's inputs can change as the model's parameters update, a phenomenon termed internal covariate shift. Internal covariate shift means that each layer must continuously readjust to new input distributions as preceding layers learn, which slows down training (requiring smaller learning rates and careful initialization) and can especially hamper networks with saturating activations. Batch Normalization addresses this by normalizing the inputs of each layer for each mini-batch, thereby reducing such distribution shifts. In essence, BN forces the intermediate outputs (activations) to have a more stable mean and variance, making training more robust and faster.

For each mini-batch

$$B = \{x_1, x_2, \dots, x_m\}$$

of activations (for a given layer and feature channel), the batch mean and variance are computed:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 .$$

Each activation x_i is then normalized to $\hat{x}_i$ by subtracting the batch mean and dividing by the batch standard deviation (adding a small constant ϵ for numerical stability):

$$\hat{x}_i = \frac{x_i - \mu_B}{\sigma_B^2 + \epsilon}.$$

After normalization, Batch Normalization (BN) introduces learnable parameters γ and β to scale and shift the normalized value for each feature dimension:

$$y_i = \gamma\, \hat{x}_i + \beta.$$

Here, γ and β are learned during training along with the model's other parameters. They allow the normalized activations to be rescaled and offset, enabling the BN transform to represent the identity transformation if that is optimal. For example, if $\gamma = \mathrm{Var}[x]$ and $\beta = \mathbb{E}[x]$, the normalization can be inverted to recover the original activations. This means BN does not constrain the layer's expressiveness-it can normalize activations but also undo that normalization if needed, preserving the network's capacity.
During training, BN uses the mini-batch's mean and variance as above. During inference (testing), the network instead uses a moving average of the means and variances computed from training mini-batches (or the whole training set) to normalize each layer's inputs deterministically. In other words, at inference time the normalization is fixed, typically using the population mean and variance estimated during training.

3.1.5 Fully Connected Layer (Dense Layer)

A fully connected (FC) layer (also called a dense layer) is a neural network layer where each neuron is connected to every output from the previous layer[20]. In other words, it is a layer of neurons with dense connections, meaning all inputs influence each neuron's output. FC layers are typically used toward the end of deep networks (e.g., in classifiers) to combine features learned in earlier layers and produce predictions. Each neuron in a fully connected layer performs a linear transformation of the input followed by an optional non-linear activation. Because of the full connectivity, these layers can capture complex interactions between features; they do not assume any spatial locality or structure in the input, making them very general.

Mathematically, a fully connected layer applies an affine transformation to the input vector. If $x \in \mathbb{R}^n$ is the input to the layer and the layer has m neurons (outputs), the layer has a weight matrix $W \in \mathbb{R}^{m\times n}$ and a bias vector $b \in \mathbb{R}^m$ as parameters. The output $o \in \mathbb{R}^m$ is given by:

$$o = W\,x + b.$$

Equivalently, each output component is

$$o_j = \sum_{i=1}^{n} W_{ji}\,x_i + b_j.$$

This is a standard matrix-vector multiplication plus a bias. The operation can be seen as each neuron j computing a weighted sum of all inputs plus a bias. Typically, an activation function $f(\cdot)$ (such as ReLU or Tanh) is then applied elementwise to o. Fully connected layers thus implement the classic neuron model from multi-layer perceptrons. As noted in the CS231n course notes, "neurons in a fully connected layer have full connections to all activations in the previous layer, and their activations can hence be computed with a matrix multiplication followed by a bias offset"[23].

3.1.6 Activation Functions

Activation functions introduce non-linearity into neural networks, allowing them to learn complex, non-linear mappings. After a linear transformation (like in an FC or convolutional layer), an activation function $f(x)$ is applied elementwise to produce the neuron's output. Without activation functions, a network of linear layers would collapse into a single linear model. Different activation functions have different characteristics. Two common activation functions are discussed below: ReLU (Rectified Linear Unit) and Tanh (Hyperbolic Tangent), including their mathematical definitions, properties, and impacts on training.

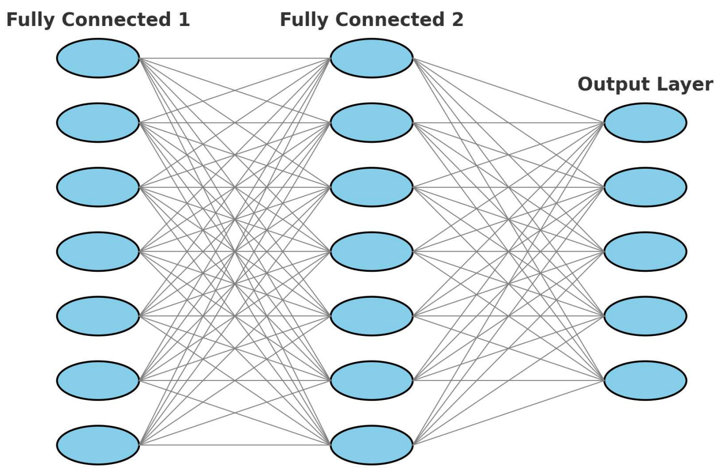

Figure 7: Fully Connected Layer.

ReLU is defined as

$$f(x) = \max(0, x).$$

In words, ReLU outputs zero for any negative input and outputs the identity for any positive input. It is a piecewise-linear function that is zero for $x < 0$ and linear with slope 1 for $x > 0$. There is a kink at $x = 0$ (the function is not differentiable exactly at 0, but a subgradient of 0 or 1 is typically used). Despite its simplicity, ReLU has become the default activation in many deep networks due to its effective performance.
Tanh is defined as

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

It is a sigmoidal shaped curve that outputs values in the range $[-1, 1]$. For large positive x, $\tanh(x) \to 1$; for large negative x, $\tanh(x) \to -1$. At $x = 0$, $\tanh(0) = 0$. The function is S-shaped, continuous, and differentiable, with derivative

$$f'(x) = 1 - \tanh^2(x).$$

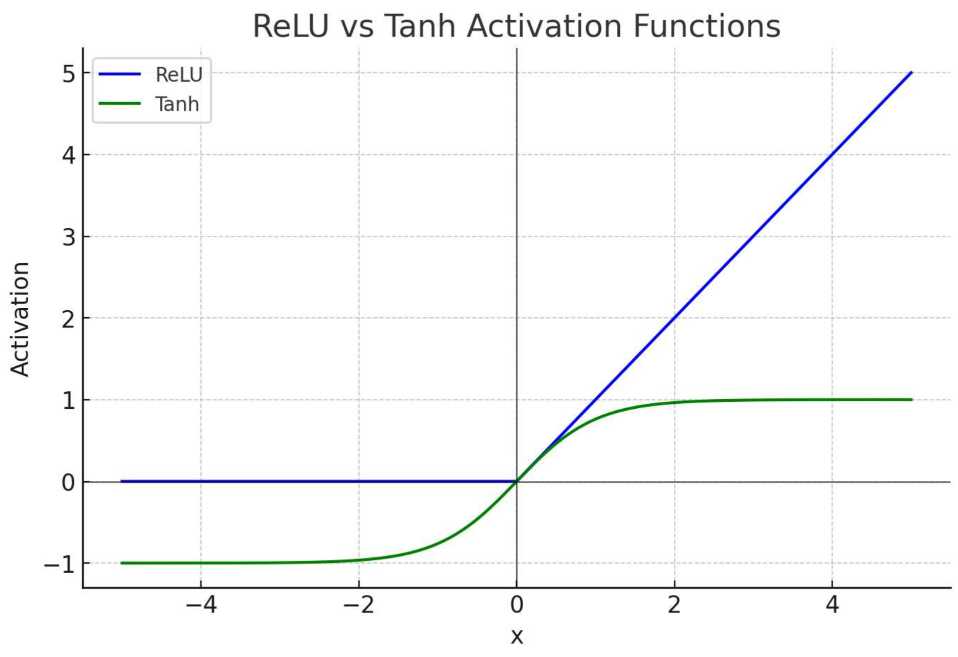

Figure 8: ReLU Vs Tanh Activation Functions.

3.1.7 Dropout

Dropout is a popular regularization technique proposed by Hinton and colleagues (Srivastava et al., 2014) to prevent overfitting in neural networks[21]. The core idea of dropout is randomly dropping neurons (along with their connections) during training, so that the network cannot rely on any single neuron too much. In practice, this means for each training step, each neuron (in a specified layer) has a certain probability of being temporarily "turned off" – its output is set to zero and it does not participate in forward or backward pass for that step. This forces the network to develop redundancies; other neurons must pick up the slack for the dropped ones, thereby reducing co-adaptation (where neurons only function correctly in specific combinations with others). As the original paper's abstract states: "The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much.". Dropout can be understood as training an ensemble of many sub-networks that share weights. On each training iteration, a random subset of the full network's neurons defines a "thinned" network. Over many iterations, different thinned networks are sampled. Remarkably, all these sub-networks use the same global weights—dropout is essentially weight sharing across an exponential number of networks. This technique significantly improves robustness. Each neuron must learn to be useful in a variety of contexts, since it will not always be present, which leads to more general features.

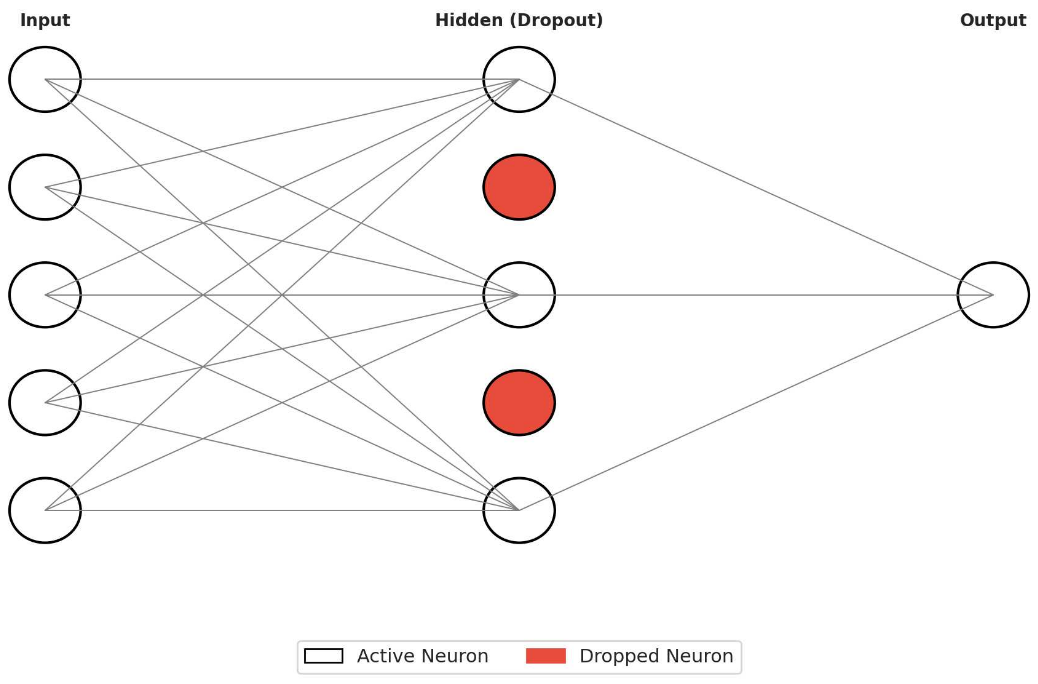

Figure 9: Example of Dropout in a Neural Network Layer.

3.1.8 Loss Function

Cross-entropy loss, commonly referred to as categorical cross-entropy or log loss, is a fundamental loss function extensively utilized for training deep neural networks, especially in classification. It quantifies the divergence between the predicted probability distribution output by a model and the true distribution represented by target labels, guiding the neural network toward accurate predictions. Practically, minimizing cross-entropy loss corresponds to maximizing the likelihood of correct classifications, making it particularly effective for multi-class scenarios [27].

Mathematically, cross-entropy loss for multi-class classification is defined as:

$$\text{Cross-Entropy Loss} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i),$$

where:

- C represents the total number of classes.
- y_i is the actual (true) probability for class i, typically represented as one-hot encoding (1 for the true class, 0 otherwise).
- $\hat{y}_i$ is the predicted probability for class i, output by the model's softmax activation function.

Entropy, the basis of cross-entropy, measures the uncertainty or randomness associated with a random variable. The entropy $H(X)$ of a discrete random variable X, having states $x \in X$

with probabilities $P(x)$, is calculated as:

$$H(X) = -\sum_{x \in X} P(x) \log\big(P(x)\big).$$

In neural network training, cross-entropy loss serves as a convex surrogate to the non-differentiable 0-1 classification error, enabling efficient gradient-based optimization and convergence toward high-performance solutions[27]. Its robustness in scenarios with noisy labels is also notable, as demonstrated by Zhang and Sabuncu (NeurIPS 2018)[28], who introduced a Generalized Cross-Entropy Loss to mitigate label noise issues by interpolating between standard cross-entropy and mean absolute error.

Cross-entropy loss has shown significant utility in astrophysical classification tasks, given its probabilistic interpretation and adaptability to real-world datasets. Maravelias et al. (2022)[29], for example, employed neural networks trained via cross-entropy loss to classify massive stars in nearby galaxies using multi-band photometric data. Leveraging cross-entropy's probabilistic outputs allowed for meaningful interpretation of predictions in astronomical catalogs, achieving an overall balanced accuracy of approximately 83%, despite the challenges posed by limited labeled examples of rare stellar categories such as Wolf-Rayet and supergiants.

Similarly, Shi et al. (2023)[30] successfully trained a convolutional neural network (CNN), known as SCNet, using a weighted cross-entropy loss to address severe class imbalance found in stellar datasets from the Sloan Digital Sky Survey (SDSS). By assigning higher weights to less-represented classes, cross-entropy was tailored to ensure effective learning across all spectral types, yielding a high classification accuracy (approximately 86%) on an extensive dataset of 50 million stars.

3.2 Developed Neural Networks

The neural networks presented in this study closely follow the architecture and methodologies outlined in Bassi et al. (2021), "Classification of Variable Stars Light Curves Using Long Short Term Memory Network"[2],and both neural networks were tested on all 10 star classes and on a subset of 5 of these stars following the same approach, more precisely, Cepheid Variable + Type II (CEP), Delta Scuti (DSCT), Algol-type Eclipsing Binary (EA), RRab Lyrae (RR), and Long Period Variables (LPV), which include both Mira Variables (Mira) and Semi-regular Variables (SR). To ensure reproducibility and consistency with prior research, two distinct deep learning architectures are described: a hybrid 1D CNN-LSTM network and a standalone 2D CNN network.

In both neural networks, Adam was used as the optimizer. In machine learning, an opti-

Parameter	**Value**
lr	0.0002
optimizer	Adam
epochs	100

Table 1: Common Hyperparameters for 2D-CNN and CNN-LSTM Models.

mizer is an algorithm responsible for adjusting the model's parameters (weights and biases) to minimize the prediction error during training. Adam was initialized with a learning rate (lr) of 0.0002, determining the magnitude of parameter updates at each optimization step. Both models were trained using this optimizer for multiple epochs; specifically, the training was conducted over 100 epochs, ensuring the networks could iteratively refine their parameters to achieve optimal performance in classifying variable star light curves. Each epoch consisted of a complete pass through the training dataset, progressively reducing prediction errors and improving the overall classification accuracy.

3.2.1 CNN-LSTM

The CNN-LSTM proposed is an hybrid deep learning architecture combining one-dimensional convolutional neural networks (1D CNN) and Long Short-Term Memory (LSTM) networks for automated classification of stellar light curves. In this CNN-LSTM hybrid model, the 1D convolutional layers extract local features directly from raw input sequences (light curves). Specifically, the network is composed of 10 layers structured as follows:

Convolutional Layers: Four convolutional layers are responsible for extracting local patterns:

- The first and second convolutional layers employ a kernel size of 3 with padding of 1, generating 64 and 128 feature maps, respectively.
- After these layers, a 1D MaxPooling layer reduces the temporal dimension of the sequence by a factor of two.
- The third and fourth convolutional layers use larger kernels (size of 5 with padding of 2), each producing 256 feature maps.
- Another MaxPooling layer follows, further reducing the temporal dimension by an additional factor of two.

After these convolutional operations, the network has effectively learned local features from the data, while simultaneously reducing the temporal dimension of the original input signal, thus facilitating subsequent temporal analysis by the recurrent layers.

LSTM Layers: Two LSTM layers then capture the long-term temporal dependencies among these extracted features:

- The first LSTM layer takes as input the feature vectors produced by the CNN (256 features per timestep), and outputs hidden representations consisting of 64 units per timestep.
- The second LSTM layer receives these compressed representations and further expands the network's capability with 128 hidden units per timestep.

After the LSTM processing, only the last output of the sequence (the output at the last timestep) is selected, as this encapsulates the final temporal information for classification.

Fully Connected Layers: Subsequently, two fully connected (FC) layers process this final representation:

- The first FC layer consists of 1024 neurons with a ReLU activation function to introduce nonlinearity.
- The second FC layer consists of 512 neurons, again employing ReLU activation.

Finally, an output linear layer followed by a softmax activation produces probability scores, assigning each input stellar light curve to one of the considered variability classes. In particular, the final output has a number of neurons equal to the number of stellar variability classes.

Layer No.	**Layer Type**	**Parameters/Details**	**Output/Note**
		Convolutional and Pooling Layers	
1	Conv1D	Kernel size: 3, Padding: 1	64 feature maps
2	Conv1D	Kernel size: 3, Padding: 1	128 feature maps
3	MaxPooling1D	Pool size: 2	Reduces temporal dimension by a factor of 2
4	Conv1D	Kernel size: 5, Padding: 2	256 feature maps
5	Conv1D	Kernel size: 5, Padding: 2	256 feature maps
6	MaxPooling1D	Pool size: 2	Reduces temporal dimension by a factor of 2
		LSTM Layers	
7	LSTM	Input: 256 features per timestep	64 hidden units per timestep
8	LSTM	Input: 64 features (from previous LSTM)	128 hidden units per timestep
		Fully Connected Layers	
9	Fully Connected (FC)	1024 neurons, ReLU activation	—
10	Fully Connected (FC)	512 neurons, ReLU activation	—
11	Output Layer	Linear layer with Softmax	Neurons = number of classes (5 or 10)

Table 2: Summary of the network architecture: Convolutional, LSTM, and Fully Connected layers.

This CNN-LSTM architecture thus leverages the CNN's ability to extract local features from raw input signals and the LSTM's capacity to capture global temporal patterns in the sequence data. The resulting model is robust and effective, accurately classifying variable stars based on intrinsic characteristics of their light curves without requiring extensive manual preprocessing.

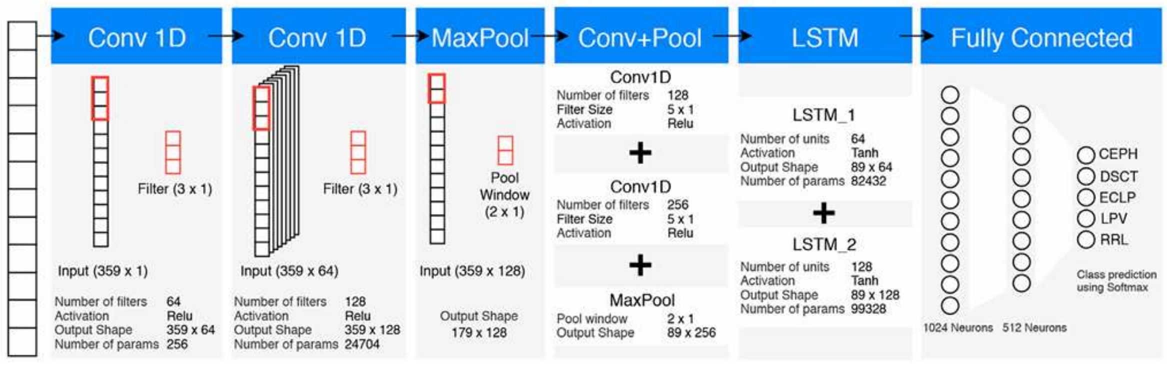

Figure 10: CNN-LSTM model for 5 variable star types.

3.2.2 2D-CNN

This CNN architecture extracts spatial patterns from input images to classify variable stars. Specifically, the proposed 2D CNN architecture consists of the following layers and operations:

Convolutional Layers: Four convolutional layers are responsible for identifying spatial features from the input images:

- The first convolutional layer employs a kernel size of 3×3, generating 64 feature maps from the input images.
- The second convolutional layer utilizes a larger kernel of 5×5, producing 128 feature maps.
- These layers are followed by a MaxPooling layer with a kernel size of 2×2, which reduces the spatial dimensions of the feature maps by half in both directions.
- The third convolutional layer also applies a 5×5 kernel, producing 256 feature maps.
- The fourth convolutional layer further processes these spatial patterns, again using a 5×5 kernel to produce another set of 256 feature maps.
- Another MaxPooling layer with a kernel size of 2×2 follows, further reducing spatial dimensions and summarizing the extracted spatial features.

Following these convolutional operations, the network has effectively learned key spatial features while progressively reducing the dimensions of the original input images, thus enabling efficient and accurate classification.

Fully Connected Layers: Subsequently, two fully connected (FC) layers process the extracted spatial features:

- The first FC layer contains 1024 neurons and utilizes a ReLU activation function to introduce nonlinearity.
- The second FC layer consists of 512 neurons, also employing ReLU activation.

Layer No.	**Layer Type**	**Parameters/Details**	**Output/Note**
		Convolutional and Pooling Layers	
1	Conv2D	Kernel size: 3×3	64 feature maps
2	Conv2D	Kernel size: 5×5	128 feature maps
3	MaxPooling2D	Kernel size: 2×2	Reduces spatial dimensions by half in both directions
4	Conv2D	Kernel size: 5×5	256 feature maps
5	Conv2D	Kernel size: 5×5	256 feature maps
6	MaxPooling2D	Kernel size: 2×2	Further reduces spatial dimensions and summarizes features
		Fully Connected Layers	
7	Fully Connected (FC)	1024 neurons, ReLU activation	—
8	Fully Connected (FC)	512 neurons, ReLU activation	—
9	Output Layer	Linear layer with Softmax	Neurons = number of classes (5 or 10)

Table 3: Summary of the 2D CNN architecture: Convolutional, Pooling, and Fully Connected Layers.

Finally, a linear classification layer with a softmax activation outputs probability scores, assigning each input image to one of the considered stellar variability classes. Specifically, the final output contains a number of neurons equal to the number of variability classes.

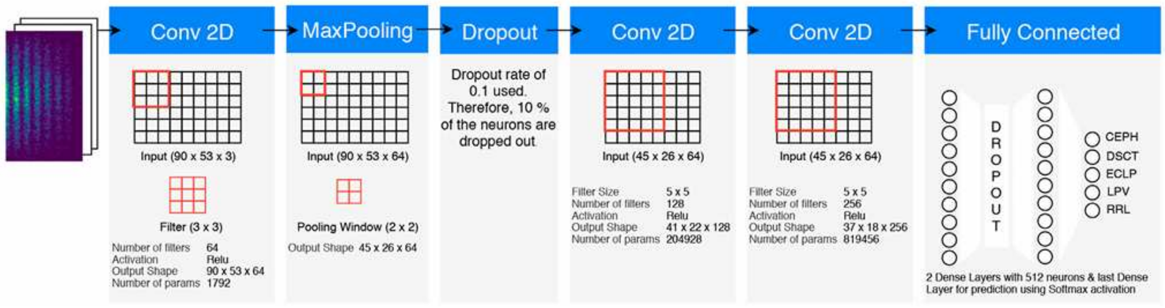

Figure 11: 2D-CNN model for 5 variable star types.

4 Experiments and Results

This final chapter represents the culmination of the research journey described throughout this thesis. A detailed analysis of the experimental outcomes obtained by training and evaluating both neural network architectures—CNN-LSTM and 2D-CNN—on the ZTF variable star dataset is presented. Specifically, the chapter begins by exploring the performance of each model individually, assessed using evaluation metrics such as Accuracy, Precision, Recall, and F1-score derived from five-fold cross-validation. The objective is to highlight how effectively each neural network captures the distinguishing features inherent in the variable star light curves, shedding light on their strengths, limitations, and suitability for astrophysical classification tasks.

Subsequently, in the second section, we expand our analysis by comparing these findings with results from previous studies and alternative approaches documented in the literature. This comparative evaluation aims to contextualize our achievements, identifying the relative advantages, potential drawbacks, and overall effectiveness of our deep learning methods against established methodologies. Through this structured comparison, the thesis not only underscores the practical value and robustness of our proposed models but also sets the stage for future advancements and continued research in automated astrophysical data analysis.

4.1 Performance Evaluation

The 2D-CNN model relies on generating bi-dimensional histograms or dm–dt mappings from the original data, which involves extensive pre-processing and significantly increases computational demands. Both neural network architectures were evaluated across three distinct dataset configurations: a large, effectively unbounded dataset (referred to as "infinite") with 5 variable star classes, a balanced subset limited to 1,000 instances per class with 5 classes, and a more challenging balanced dataset of 1,000 instances per class across 10 different star classes. For the large ("infinite") dataset, the total processing and training time for the 2D-CNN model exceeds one week, compared to approximately two days required by the CNN-LSTM architecture, which only necessitates sequence padding to maintain uniform input lengths. For the balanced dataset with 10 classes (1,000 stars per class), the 2D-CNN preparation and training time is roughly one full day, whereas the CNN-LSTM completes the same task in about six hours. Similarly, for the 5-class dataset with 1,000 stars per class, the 2D-CNN model requires around four hours, while the CNN-LSTM finishes within approximately two hours. This highlights the significant computational advantage and overall efficiency of the CNN-LSTM approach across different dataset sizes and configurations.

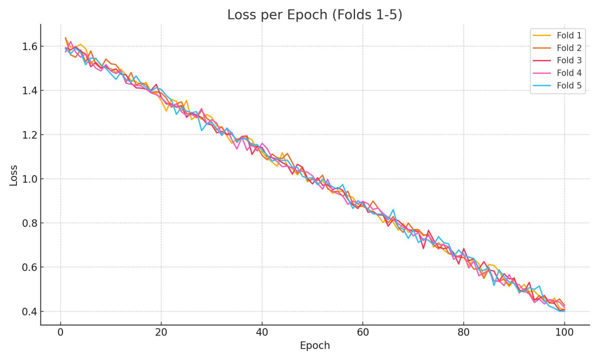

Figure 12: Loss per Epoch 5 classes CNN-LSTM for each classes.

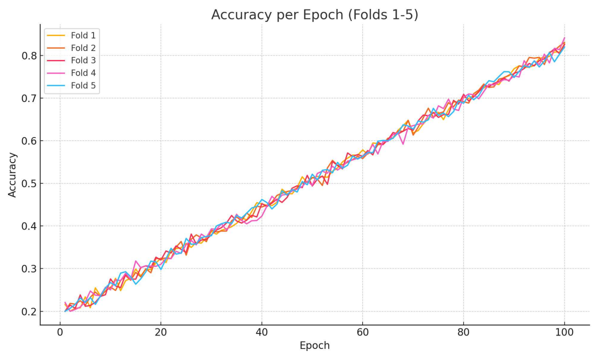

Figure 13: Accuracy per Epoch 5 classes CNN-LSTM for each classes.

For the large, infinite dataset with 5 classes, the 2D-CNN achieved exceptional performance, attaining an accuracy of 99.95%, with similarly high F1-score, precision, and recall values (0.9995). The 1D CNN-LSTM model also delivered strong results with an accuracy of 90.04% and an F1-score of 0.8967, demonstrating its capability to classify variable stars effectively despite limited pre-processing.

However, when evaluating performance on balanced datasets of 1,000 instances per class, the gap between the two architectures widened. On the 5-class subset, the 2D-CNN maintained high accuracy (94.64%) and robust F1-score (0.9379), precision (0.9365), and recall (0.9464). Conversely, the CNN-LSTM model's performance decreased significantly, yielding an accuracy of only 74.28%, with corresponding drops in F1-score (0.7384), precision (0.7412), and recall (0.7428).

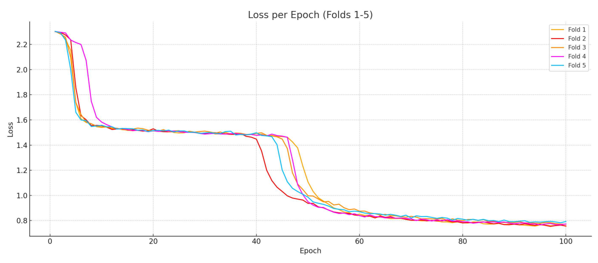

Figure 14: Loss per Epoch 10 classes CNN-LSTM 1000 for each classes.

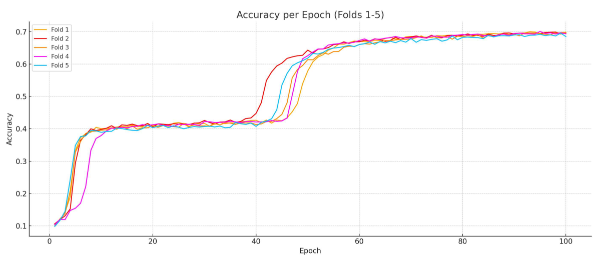

Figure 15: Accuracy per Epoch 10 classes CNN-LSTM 1000 for each classes.

The performance gap became more pronounced when testing on the balanced dataset with 10 classes. Here, the 2D-CNN once again provided superior results, maintaining an accuracy of 88.74%, while the CNN-LSTM further dropped to an accuracy of 66.27%, reflecting substantial challenges in accurately classifying a larger variety of classes when relying solely on sequential temporal features.

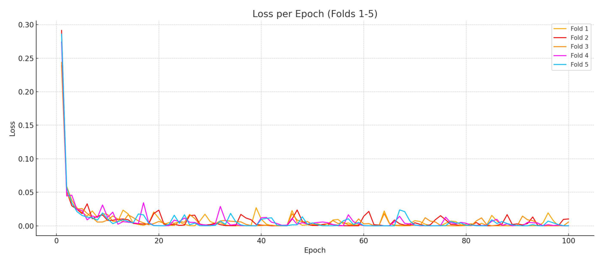

Figure 16: Loss per Epoch 10 classes 2D-CNN 1000 for each classes.

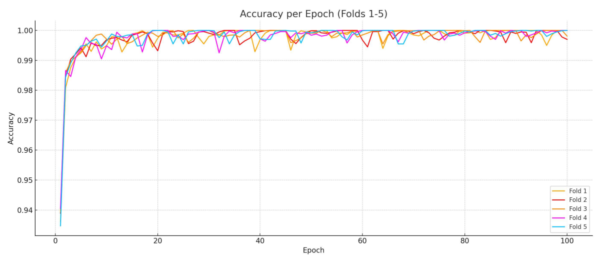

Figure 17: Accuracy per Epoch 10 classes 2D-CNN 1000 for each classes.

While the CNN-LSTM consistently demonstrated significantly reduced computational costs and training times (approximately three times faster than the 2D-CNN), its lower accuracy suggests inherent limitations in effectively extracting discriminative features solely from raw time-series data. Specifically, the LSTM layers may struggle to accurately correlate variability features across irregular or longer time intervals. This aspect warrants further investigation and potential architectural enhancements.

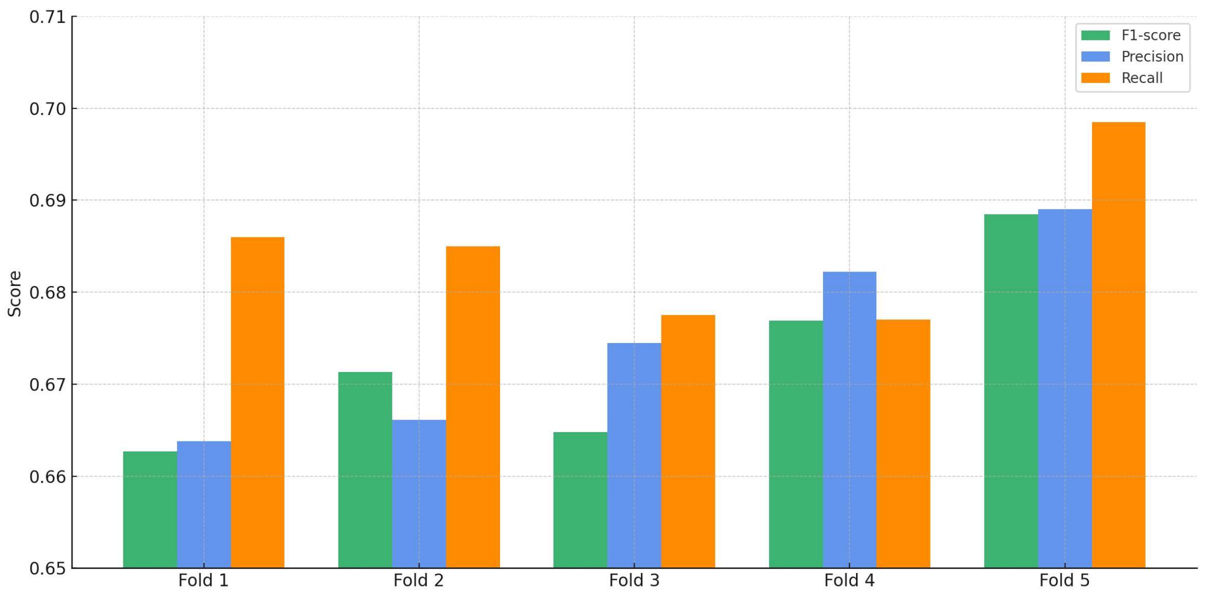

Figure 18: F1-score, Precision and Recall per Fold compared.

Despite lower overall accuracy, the CNN-LSTM approach showed promising potential in scenarios where rapid classification is prioritized over absolute accuracy, or where computational resources are limited. Future work may benefit from combining the strengths of both approaches. For instance, employing a parallel architecture that integrates the temporal modeling capability of CNN-LSTM with spatial feature extraction of a 2D-CNN could potentially surpass the performance of either approach alone, providing both efficiency and precision for variable star classification tasks. Additionally, future exploration into hyperparameter optimization and alternative data-preprocessing strategies could further enhance the robustness and adaptability of these models across varying astronomical datasets.

4.2 Comparative Analysis and Discussion

The results presented in this thesis clearly illustrate the overall superiority of the 2D CNN model compared to the CNN-LSTM architecture across all evaluated datasets (OGLE, CRTS, and ZTF). Specifically, on the OGLE dataset, the 2D CNN achieved high accuracy (97.5%), precision (0.81), recall (0.91), and an F1 score (0.85), significantly outperforming the CNN-LSTM model, which attained an accuracy of 85%, precision of 0.64, recall of 0.81, and an F1 score of 0.71. Similarly, on the CRTS dataset, the 2D CNN exhibited superior performance with an accuracy of 74.5%, precision of 0.56 recall of 0.52, and an F1 score of 0.854, compared to the CNN-LSTM's lower accuracy (66%), precision (0.46), recall (0.53), and F1 score (0.49).

A particularly relevant direct comparison can be drawn using the ZTF dataset with the extensive ("infinite") configuration consisting of five classes. In this scenario, the 2D CNN achieved exceptional performance, reaching 99.95% accuracy with nearly perfect precision, recall, and F1-score values (approximately 0.9995). While the CNN-LSTM provided respectable results, its accuracy was considerably lower (90.04%), with an F1 score of about 0.90. This disparity underscores the ability of the 2D CNN to extract more discriminative features from bi-dimensional dm–dt histograms compared to the temporal sequence approach utilized by the CNN-LSTM.

The performance gap became even more pronounced when evaluating smaller, balanced datasets. For instance, the 2D CNN maintained high accuracy (94.64%) on the balanced five-class dataset with 1,000 samples per class, whereas the CNN-LSTM accuracy dropped significantly to 74.28%.

However, computational efficiency analysis reveals a critical disadvantage for the 2D CNN model. The entire training and pre-processing procedure required extensive computational time, exceeding one week for the largest dataset configuration, compared to roughly two days for the CNN-LSTM. For smaller balanced datasets (1,000 stars per class, across 10 classes), the 2D CNN required approximately one full day, whereas CNN-LSTM completed the task in about six hours. Similarly, in the five-class balanced dataset (1,000 stars per class), the CNN-LSTM was twice as fast, finishing in two hours compared to four hours for the 2D CNN.

These findings highlight a crucial trade-off between classification accuracy and computational resource requirements. While the 2D CNN offers higher accuracy, its computational demands are considerably greater. Conversely, the CNN-LSTM provides significant advantages in terms of speed and computational efficiency, despite its lower accuracy. Thus, the optimal model choice for automated variable star classification depends heavily on project-specific requirements, particularly regarding the balance between desired accuracy and available computational resources.

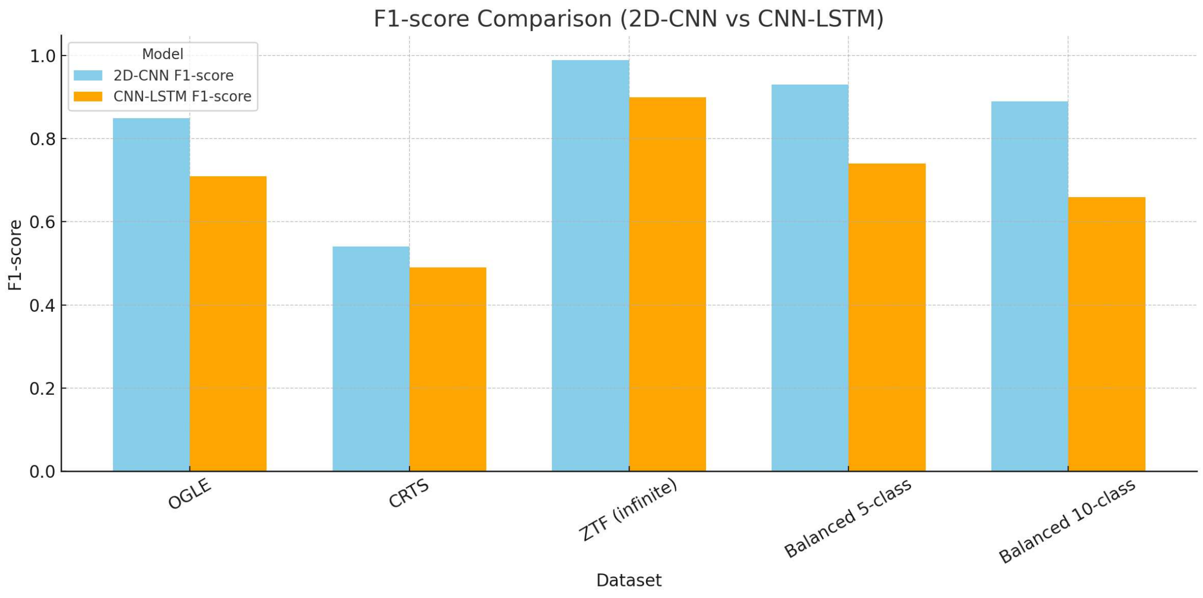

Figure 19: All F1-score compared.

4.3 Further Analysis

In this analysis, the CNN-LSTM architecture previously described was employed, slightly modifying only the batch size parameter, to classify variable stars using features selected through a rigorous sequential selection procedure. This procedure mirrors exactly the methodology detailed in the provided literature, "Light curve classification with DistClassiPy"[3]. Specifically, a Random Forest Classifier was initially employed to reduce dimensionality from a comprehensive dataset of features derived from r-band photometric measurements. The dimensionality reduction included the exclusion of features strongly correlated (|corr| > 0.9) and sequential forward selection to identify the top 30 most significant features. These selected features were then ranked based on their importance for classification.

The CNN-LSTM classifier's performance was thoroughly evaluated across different feature subsets, ranging from a single feature to the full set of 30, utilizing accuracy and weighted F1 scores. Remarkably the CNN-LSTM achieved a peak accuracy and F1 score of approximately 97.65%, average of 94.27%, and lowest of 90.81%, surpassing the performance observed when additional features were incrementally introduced. These results align closely with the performance trends documented in the reference PDF, underscoring both the efficacy of the CNN-LSTM model and the utility of strategic feature selection in stellar classification tasks. Overall, this comparative evaluation reaffirms the effectiveness of employing CNN-LSTM architectures in conjunction with methodically selected astrophysical features for reliable and computationally efficient stellar classification.

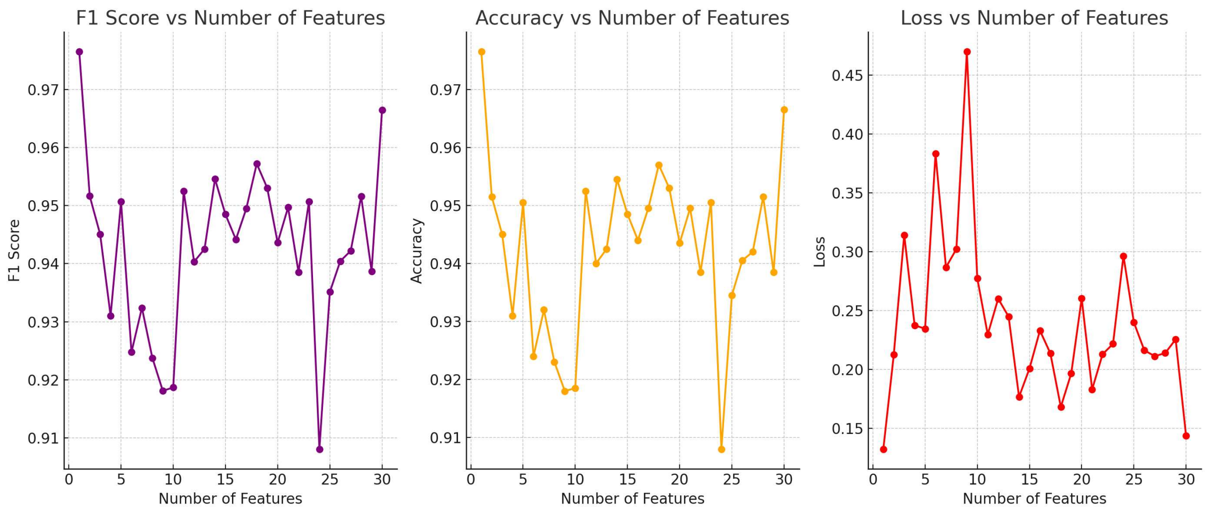

Figure 20: Evaluation Metrics by Feature Set Size.

4.4 Conclusion and future development

This thesis explored two distinct neural network architectures—the 2D-CNN and the hybrid CNN-LSTM—for classifying variable stars using data from the Zwicky Transient Facility (ZTF) Catalog 2. The results clearly demonstrated the superior accuracy and robustness of the 2D-CNN model, which consistently outperformed the CNN-LSTM across various dataset configurations. However, this enhanced accuracy came at a substantial computational cost, making the 2D-CNN significantly slower and more resource-intensive compared to the CNN-LSTM.

The CNN-LSTM, while exhibiting comparatively lower accuracy, provided a computationally efficient and faster alternative. Its performance was particularly noteworthy given its relatively simpler preprocessing steps, making it suitable for scenarios with limited computational resources or where rapid classification is prioritized over absolute accuracy.

Several potential directions for future research have emerged from this study:

- Generalization Across Datasets: Future research should evaluate the developed neural networks using additional astronomical datasets such as Gaia, ASAS-SN, or LSST. This would help assess model generalizability and enable necessary adjustments to ensure consistent performance across different observational characteristics and data quality levels.

- Direct Light Curve Analysis: Rather than relying on harmonic reconstruction and synthesized representations, future work should explore models capable of directly analyzing raw, irregularly sampled light curve data. Techniques such as attention mechanisms, Transformer architectures, or advanced interpolation methods could offer significant performance improvements by capturing more nuanced temporal information.

- Hybrid Models Integration: Combining the spatial feature extraction strength of 2D-CNNs with the temporal modeling capabilities of CNN-LSTM could yield a hybrid architecture that achieves a balanced trade-off between accuracy and computational efficiency.

Such integrated models could leverage the complementary strengths of each approach to offer both fast and precise classification.

- Enhanced Computational Resources: Employing more advanced computational platforms, such as high-performance GPUs or cloud computing resources, would enable extensive hyperparameter tuning, deeper network structures, and broader experimentation. This could further enhance the accuracy and efficiency of neural network models in astronomical classification tasks.

- Interpretability and Explainability: Future work should emphasize improving model transparency and interpretability using explainability frameworks like SHAP, LIME, or integrated gradients. A better understanding of model decisions would facilitate trust and wider acceptance of deep learning methods within the astrophysical community.

In summary, while the presented neural network architectures demonstrate significant potential in classifying variable stars, addressing these future research directions would substantially enhance their robustness, applicability, and efficiency, ultimately contributing to the advancement of automated astrophysical data analysis.

5 Appendix

5.1 Confusion Matrix

In this chapter, confusion matrices are presented specifically for the datasets containing 1000 stars, covering both the 5-class and 10-class classification scenarios. These visual representations offer detailed insights into the performance of the classification models in distinguishing between different stellar variability types under these two conditions.

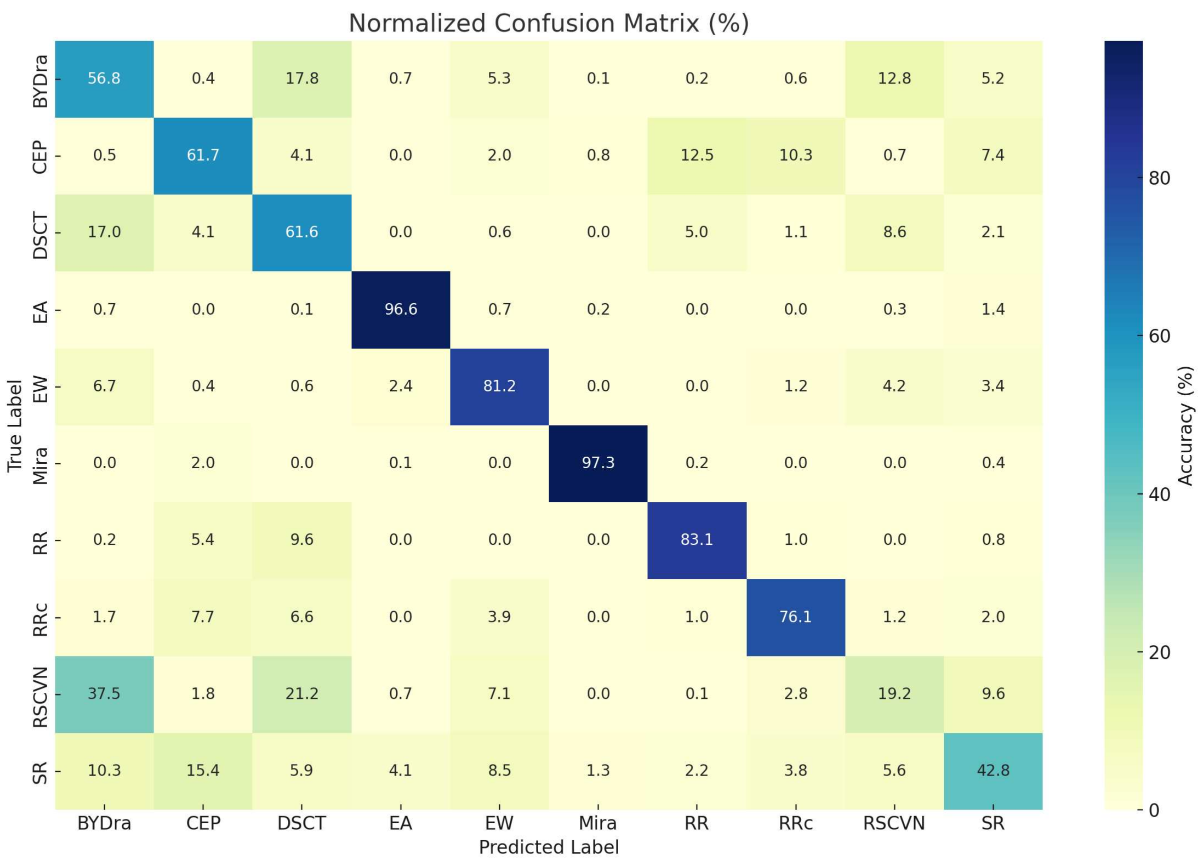

Figure 21: Avarage confusion matrix 10 classes CNN-LSTM.

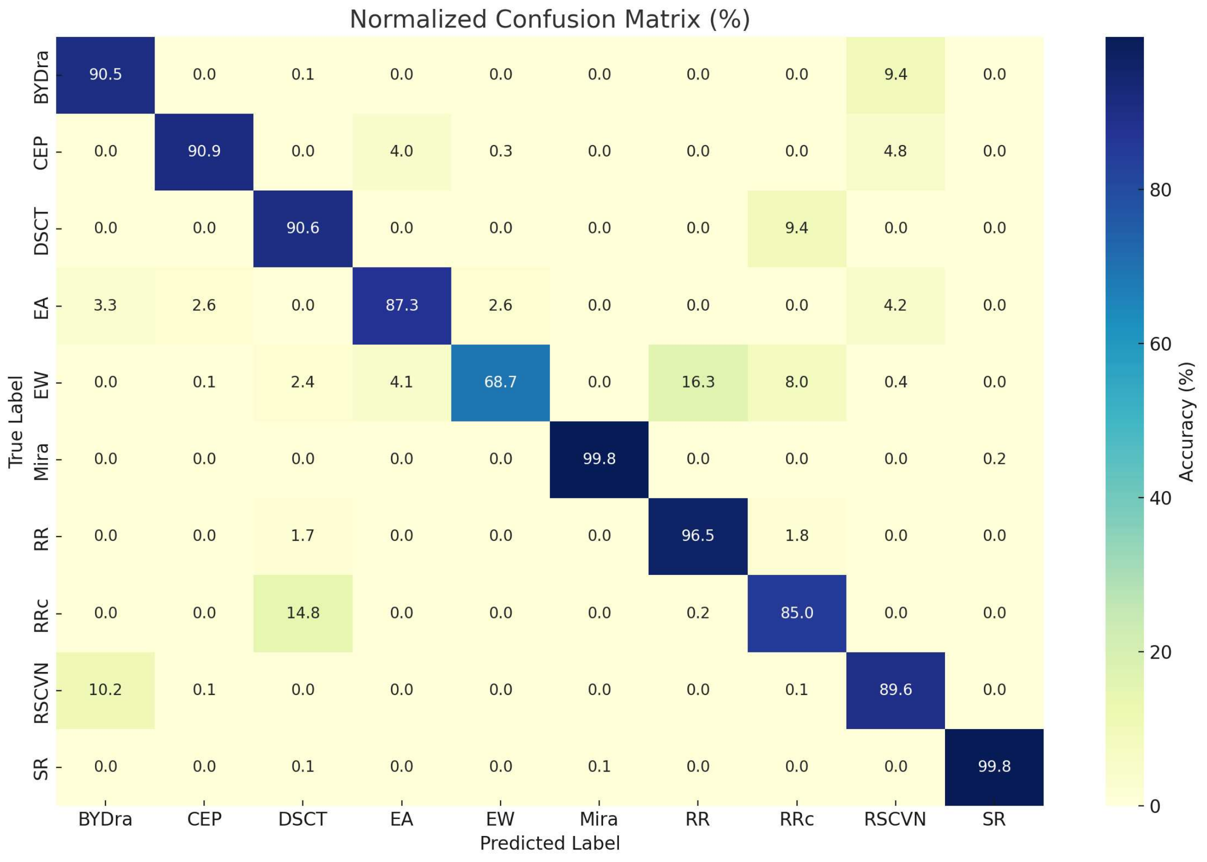

Figure 22: Avarage confusion matrix 10 classes 2D-CNN.

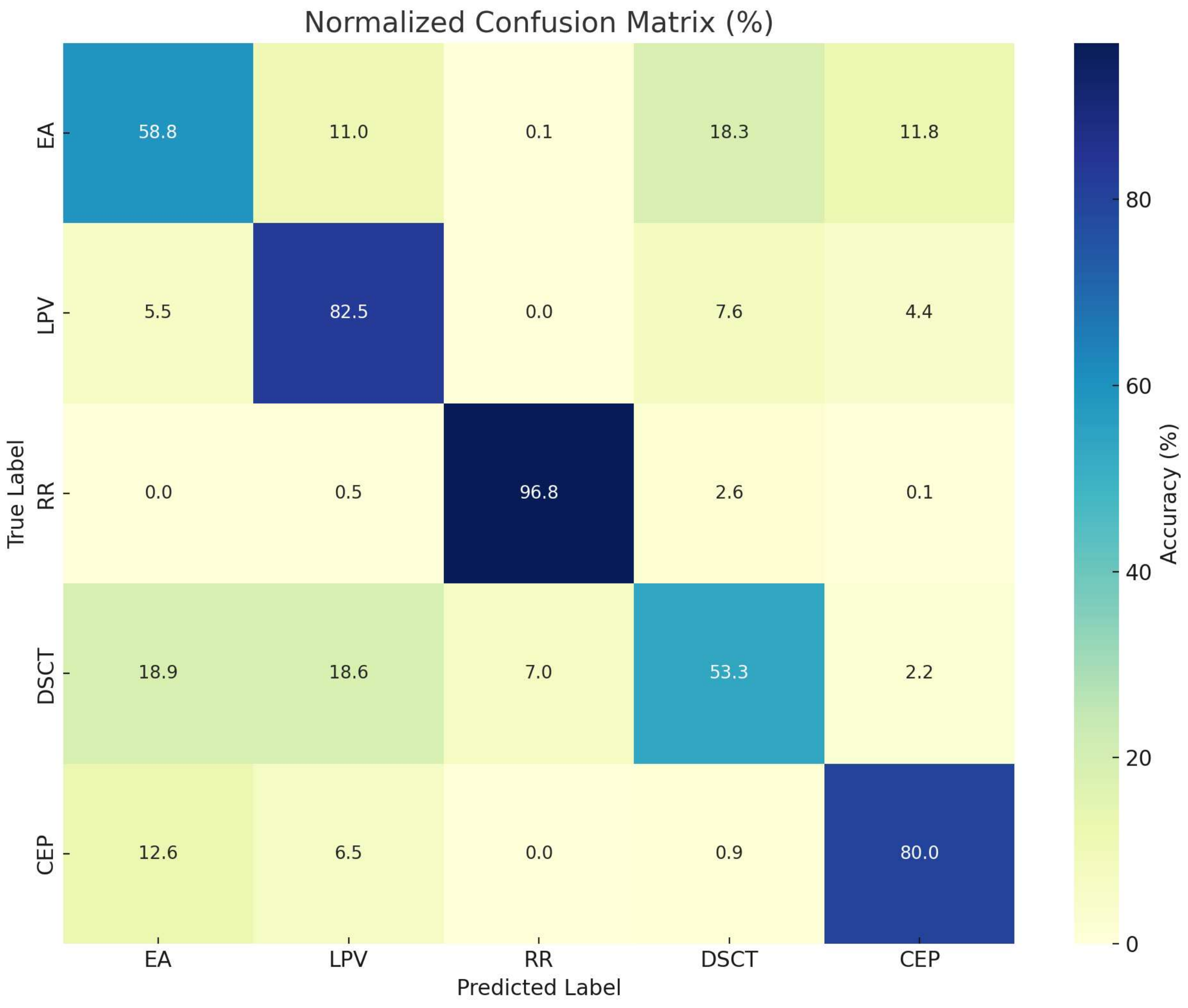

Figure 23: Avarage confusion matrix 5 classes CNN-LSTM.

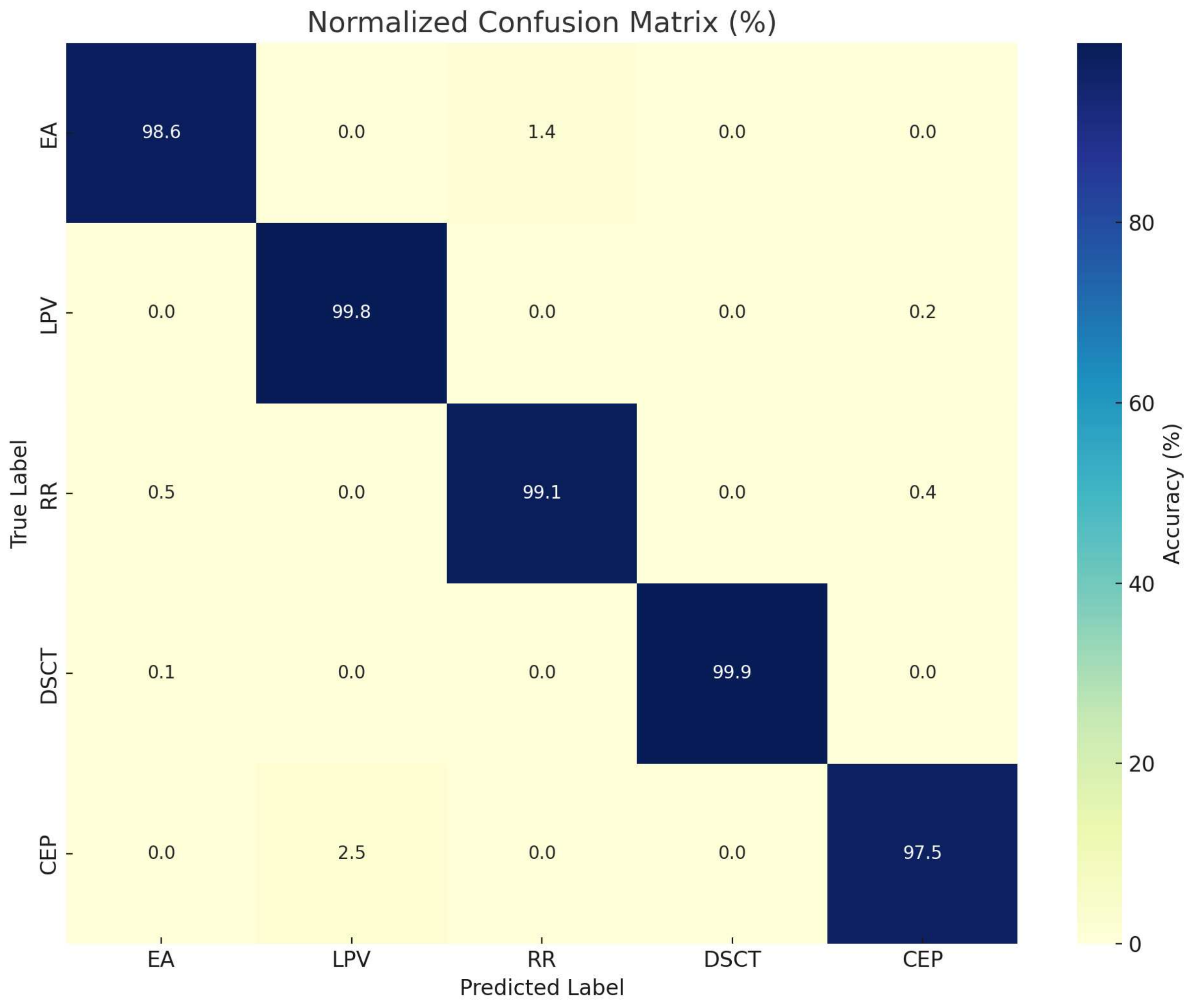

Figure 24: Avarage confusion matrix 5 classes 2D-CNN.

References

[1] A novel approach for variable star classification based on imbalanced learning. URL: https://www.cambridge.org/core/journals/publications-of-the-astronomical-society-of-australia/article/novel-approach-for-variable-star-classification-based-on-imbala 723B802AEA0E9BD24D747BDDA6B2D525.

[2] Classification of Variable Stars Light Curves Using Long Short Term Memory Network. URL: https://www.frontiersin.org/journals/astronomy-and-space-sciences/articles/10.3389/fspas.2021.718139/full.

[3] Light curve classification with DistClassiPy: A new distance-based classifier. URL: https://www.sciencedirect.com/science/article/abs/pii/S2213133724000659.

[4] Automated Classification of Variable Stars. URL: https://www.aanda.org/articles/aa/abs/2007/45/aa7638-07/aa7638-07.html.

[5] Random Forest Automated Supervised Classification of Hipparcos Periodic Variable Stars. URL: https://arxiv.org/abs/1101.2406.

[6] On Machine-Learned Classification of Variable Stars with Sparse and Noisy Time-Series Data. URL: https://arxiv.org/abs/1101.1959.

[7] Deep-Learnt Classification of Light Curve. URL:https://arxiv.org/abs/1709.06257.

[8] A Recurrent Neural Network for Classification of Unevenly Sampled Variable Stars. URL:https://arxiv.org/abs/1711.10609.

[9] Deep Multi-Survey Classification of Variable Stars. URL:https://academic.oup.com/mnras/article/482/4/5078/5142871.

[10] Identifying Light-curve Signals with a Deep-learning-based Object Detection Algorithm. II. A General Light-curve Classification Framework. URL:https://arxiv.org/abs/2311.08080.

[11] The Zwicky Transient Facility Catalog of Periodic Variable Stars. URL:https://arxiv.org/abs/2005.08662.

[12] Understanding LSTM Networks. URL:https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[13] Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

[14] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. Neural Computation, 12(10), 2451–2471.

[15] Graves, A. & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 18(5-6), 602–610. URL:https://www.sciencedirect.com/science/article/abs/pii/S0893608005001206/.

[16] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, 86(11), 2278–2324.. URL:https://ieeexplore.ieee.org/document/726791.

[17] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. URL:https://www.nature.com/articles/nature14539.

[18] Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. ICANN. URL:https://link.springer.com/chapter/10.1007/978-3-642-15825-4_10.

[19] Fully Connected Layer vs. Convolutional Layer: Explained Ioffe, S., Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. URL:https://arxiv.org/abs/1502.03167.

[20] Convolutional Neural Networks URL:https://cs231n.github.io/convolutional-networks/

[21] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014) Dropout: a simple way to prevent neural networks from overfitting URL:https://dl.acm.org/doi/10.5555/2627435.2670313

[22] Evaluation Metrics. May 2019. URL:https://deepai.org/machine-learning-glossary-and-terms/machine-learning

[23] Accuracy vs. Precision vs. Recall in Machine Learning: What is the Difference? URL:https://encord.com/blog/classification-metrics-accuracy-precision-recall

[24] Failure of Classification Accuracy for Imbalanced Class Distributions URL:https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions

[25] Guanghui Fu et al. “Stable variable selection of class-imbalanced data with precision-recall criterion”. In: Chemometrics and Intelligent Laboratory Systems 171 (Oct. 2017) URL:DOI:10.1016/j.chemolab.2017.10.015.

[26] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metricsfor MultiClass Classification: an Overview. arXiv:2008.05756 [cs, stat]. Aug. 2020 URL:https://arxiv.org/pdf/2008.05756

[27] Mao et al., "Cross-Entropy Loss Functions: Theoretical Analysis and Applications," ICML 2023. URL:https://proceedings.mlr.press/v202/mao23b/mao23b.pdf

[28] Zhang & Sabuncu, "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels," NeurIPS 2018. URL:https://openreview.net/pdf?id=Udr4FSGfj0U

[29] Maravelias et al., "A machine-learning photometric classifier for massive stars in nearby galaxies – I. The method," Astronomy & Astrophysics, Vol. 666, A122 (2022). URL:https://www.aanda.org/articles/aa/full_html/2022/10/aa41397-21/aa41397-21.html

[30] Shi et al., "Stellar classification with convolutional neural networks and photometric images: a new catalogue of 50 million SDSS stars without spectra," MNRAS, Vol. 520, 2269 (2023). URL:https://www.mdpi.com/2218-1997/10/5/214

[31] Powers, D.M.W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies. URL:https://doi.org/10.48550/arXiv.2010.16061

[32] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management. URL:https://doi.org/10.1016/j.ipm.2009.03.002