VSB TECHNICAL UNIVERSITY OF OSTRAVA
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Big Data Processing by Means of Unconventional Algorithm

Zpracování rozlehlých dat s využitím nekonvenčních algoritmů.

Ing. David Andrešič

PhD Thesis

Supervisor: doc. RNDr. Petr Šaloun, Ph.D.

Ostrava, 2021

Abstrakt a přínos práce

Během posledních let se objevilo několik algoritmů pro klasifikaci časových řad z různých oblastí. Avšak astronomické časové řady (známé také jako světelné křivky) jsou pro klasifikaci poněkud větší výzvou. Vyznačují se velkými rozdíly v délkách, periodách, zašuměnosti a nemají jasné hranice mezi jednotlivými třídami. V této práci popisujeme některé z těchto problémů na třech veřejně dostupných datových sadách a několika obecně úspěšných klasifikačních algoritmech. Také ukazujeme náš postup, který zahrnuje použití umělých neuronových sítí vylepšených evolučními algoritmy za účelem nalezení modelu s největší přesností klasifikace předzpracovaných světelných křivek s extrahovanými atributy. Náš postup je schopen konkurovat současným řešením pro tyto datové sady.

Klíčová slova

umělé neuronové sítě, vícevrstvý perceptron, astronomické časové řady, světelné křivky, long short-term memory, velká data, klasifikace

Abstract and Contributions

During last years, several successful algorithms emerged for classification of time series from various areas. But astronomical time series (a.k.a. light curves) are a bit more challenging to classify. They greatly vary in lengths, periods, noisiness and do not have clear borders between classes. In this work, we depict these issues on three publicly available data sets and several well-performing algorithms. We also present our approach that includes the use of artificial neural networks enhanced by evolutionary algorithm to find the best performing classification algorithm for pre-processed light curves with extracted features. Our approach is able to challenge results of related work that includes these data sets.

Keywords

artificial neural network; multi-layer perceptron; astronomical time series; light curves; long short-term memory; big data; classification

Acknowledgement

I would like to thank my supervisor doc. RNDr. Petr Šaloun, Ph.D. for his mentoring and scientific lead. I would also like to thank our colleague Bc. Bronislava Pečíková from Slovak University of Technology for her cooperation during her work on master thesis. Special thanks go to Dr. Eamonn Keogh of University of California Riverside for his points regarding Dynamic Time Warping. Lastly and most certainly not leastly I would like to thank my wife Radka for her devoted support and my daughter Natálie for being merciful.

Contents

List of symbols and abbreviations **7**

1 Introduction **9**
- 1.1 Types of Stars Variability . . . 9
- 1.2 Goals . . . 10

2 State of the Art **12**
- 2.1 Time Series Classification Methods . . . 12
- 2.2 Other Related Work . . . 19

3 Data Sets **20**
- 3.1 BRITE . . . 21
- 3.2 Kepler . . . 21
- 3.3 Kepler K2 . . . 22

4 Artificial Neural Networks Approaches **24**
- 4.1 Data Pre-processing . . . 25

5 Binary Classification Experiments and Results **26**
- 5.1 BRITE Data Set . . . 26
- 5.2 Kepler K2 Data Set . . . 27
- 5.3 Kepler Data Set . . . 37

6 Multi-class Classification Experiments and Results **39**
- 6.1 Data Pre-processing . . . 39
- 6.2 Experiments . . . 40

7 Case Study: Detection of *Nova Cas 2021* **58**
- 7.1 Possible Practical Use . . . 59

8 Conclusions **64**

Bibliography **66**

A List of Publications **71**
A.1 Author's Bibliography Related to the Thesis 71

Bibliography **71**
A.2 Author's Bibliography Unrelated to the Thesis 72

Bibliography **72**

B Author's Curriculum Vitae **73**
B.1 Education 73
B.2 Work Experience 74
B.3 Certificates 76
B.4 Computer skills 77
B.5 Language Skills 77

List of symbols and abbreviations

ANN – Artificial Neural Network
K2SFF – K2 Extracted Lightcurves
MLP – Multi-layer Perceptron
CNN – Convolutional Neural Network
FT – Fourier Transform
FFT – Fast Fourier Transform
PSO – Particle Swarm Optimization
BRITE – BRIght Target Explorer
LSTM – Long Short-term Memory
COTE – Collective of Transformation-based Ensembles
HIVE-COTE – Hierarchical Vote Collective of Transformation-based Ensembles
FCN – Fully-Convolutional Network
DTW – Dynamic Time Warping
BOSS – Back of SFA symbols
KNN – K-Nearest Neighbour
SVM – Support Vector Machine
RF – Random Forest
SVML – SVM with linear kernel
SVMQ – SVM with quadratic kernel
RandF – Random Forest
RotF – Rotation Forest
ED – Euclidian Distance
BN – Bayesian Network
NB – Naive Bayes
RNN – Recurrent Neural Network
GAP – Global Average Pooling
ResNet – Residual Neural Network
MPCE – Mean Per-Class Error

TN – True Negative
TP – True Positive
FITS – Flexible Image Transport System
ML – Machine Learning
GCVS – General Catalogue of Variable Stars
NASA – National Aeronautics and Space Administration
MCDCNN – Multi Channel Deep CNN
CUDA – Compute Unified Device Architecture
ASAS – The All Sky Automated Survey
2MASS – The Two Micron All-Sky Survey

Chapter 1

Introduction

Time series classification in astronomy can be useful in many areas. In this work, we focused on so called light curves: flux or magnitude values measured in some (often irregular) period of time. This can be used to detect variable stars (of many physical classes), extra-solar planets or maybe (in future with more powerful astronomical hardware) extra-solar moons orbiting planets, stellar systems analysis and other physical phenomena.

Since these data comes from various automatized astronomical devices that produce huge amount of data (even petabytes per night [1]), we are facing big data issues. To handle a classification in this amount of data, an automated solution is crucial. In this work, we tested different variations of algorithms based on artificial neural network as a mean of machine learning to test how they perform with light curves. We also propose a network enhanced by evolutionary algorithm capable to challenge current top results on similar data sets.

1.1 Types of Stars Variability

In the Universe, most of stars are actually variable in their luminosity. Describing physical reasons of this variability is beyond the scope of this work, so we offer at least a brief overview of classes used in our training data:

- *Delta Scuti* - young pulsating stars that are similarly as *cepheids* used as a so called *standard candle* to measure distances between galaxies.

- *Detached Eclipsing Binary* - when both companions in a binary star system have no significant gravitational affect to each other.

- *Semi-Detached/Contact Eclipsing Binary* - when one of the companions in a binary star system is affected by gravitational pull of the other one (but not vice versa) which actually transfers its gas to itself (accretion).

- *Gamma Dor* - young pulsating stars with not very clear physical cause.
- *RR Lyrae ab* - pulsating stars typically with a half of mass of our Sun used as *standard candles*.
- *Other Periodic/Quasi-Periodic.*

In our training data, other stars are labeled as noise. An example of how such time series for these classes looks like is depicted on Fig. 1.1. On x axis there is always a time (usually some variation of *Julian Date*). On y axis there is usually a flux defined as the total amount of energy that crosses a unit area per unit time[1] (see Equation 1.1).

$$F = \frac{L}{4\pi r^2} \tag{1.1}$$

Where F is the flux at distance r, L is the luminosity of the source star and r is the distance between Earth and the source star.

1.2 Goals

Giving the need of automated solution due to huge amount of incoming data and current state of the art in the area of light curves classification, we identified following goals:

- Find a variable star classifier (at least a binary one) capable to challenge current state-of-the-art accuracies.
- Identify best-performing artificial neural network topologies and data pre-processing approaches.
- Verify robusness of the found approach on other light curves and astronomical time series.

---

[1] According to *COSMOS - The SAO Encyclopedia of Astronomy*: `http://astronomy.swin.edu.au/cosmos/F/Flux`

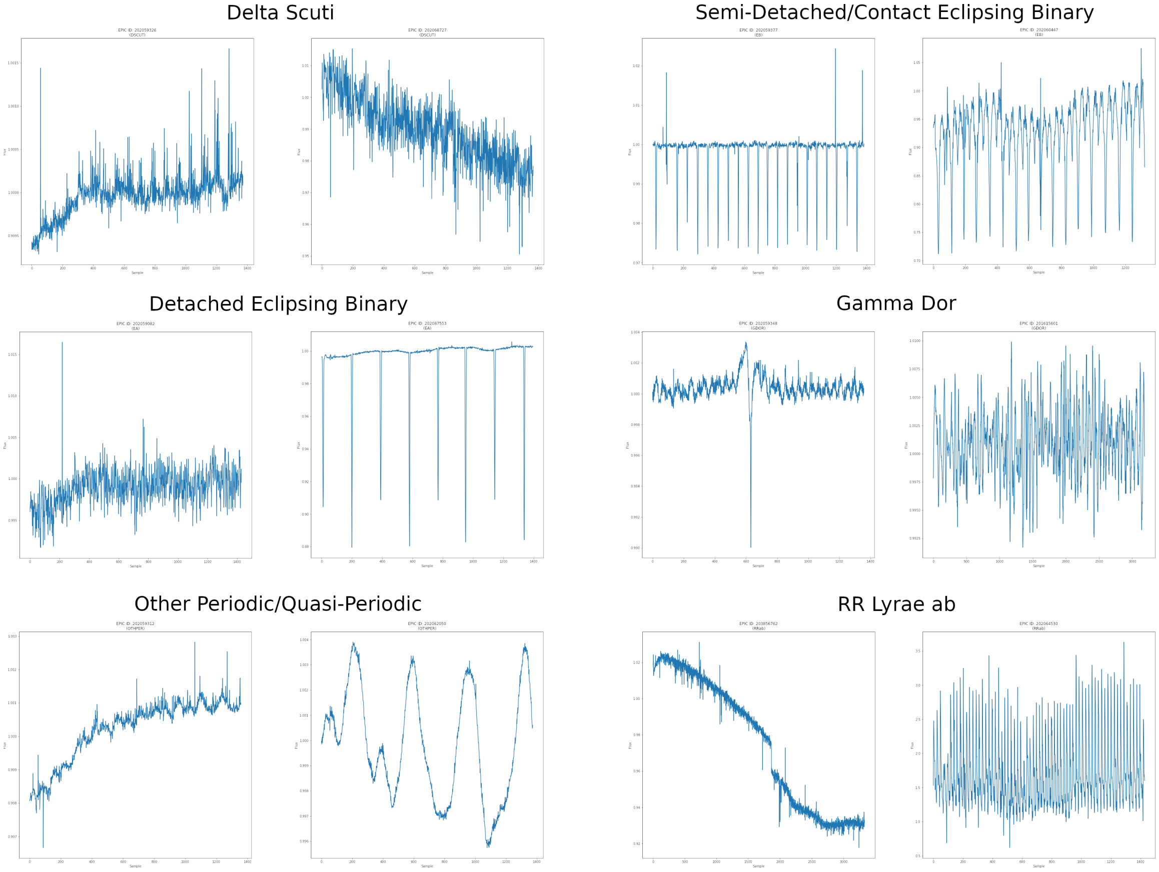


Figure 1.1: Example light curves from Kepler K2 data set. Left one from the pair is always with a low confidence in class (under 50%), right ones have high confidence in class (over 90%).

Chapter 2

State of the Art

Time series classification is a bit specific because the classifier works with sorted data where even in the order of the data a significant markers for some class can be hidden. Most literature on time series classification assumes following [2] [3]:

- copious amounts of perfectly aligned atomic patterns can be obtained,
- the patterns are all of equal length,
- every item that we attempt to classify belongs to exactly one of our well-defined classes.

For such time series, the classification using Nearest Neighbor algorithm with the relatively expensive Dynamic Time Warping as a distance measure function is usually performs best as a machine learning approach [4]. But these assumptions are challenging in astronomical time series since they greatly vary in lengths, periods, noisiness and are without clear borders between classes. In [5] authors also concludes that Long Short-Term Memory is another state-of-the-art technique for this task.

2.1 Time Series Classification Methods

During last years, several successful machine learning methods emerged for time series classification. They are often bench-marked using UCR Time Series Archive [6] made in 2002 by University of California. It is a set of data sets from different domains that is continuously extended and today it contains 128 data sets. Over 1000 papers were published using this archive until now, but it may not be so conclusive since it heavily depends on experiment configuration details [7]. Nevertheless, authors in [8] shown that on this data set COTE and HIVE-COTE performs best (followed by ResNet and FCN).

2.1.1 Traditional and Other Machine Learning Methods

Although we focused on artificial neural networks, we also comes with a brief overview of current methods and algorithms without them to summarize current state-of-the-art approaches.

2.1.1.1 Dynamic Time Warping (DTW)

This algorithm was introduced in 1968 [9] for speech recognition. It measures a distance between two time series so it can be used with K-nearest neighbor. Multiple variations of this algorithms has emerged since then, but with only marginal improvement of its original results [10]. It is still subject of ongoing research - recently it was for example migrated to GPU for speeding up and discovered that it is not ideal to apply universal phasing for aligning light curves [11]. The greatest advantage of DTW is that in opposite to a traditional Euclidean distance metric it uses a mapping between two structurally similar points which makes this metric independent [12]. It should therefore by able to identify stars of the same variability class even in case when the stars were not in the same time in the same phase. Another advantage should be the ability to discover a similarity even with different dense of measurements (or speed of change) which means that it could discover a similarity of time series of two variable stars with different gaps between measurements ("sample rate"). It also means that it should be able to discover a similarity even in case of different periods. Authors in [7] compares similar methods of time series classification and it turned out that they did not achieved significantly better results than DTW.

Since we have labeled data, we prefer a supervised learning, which is why we did not considered this method. The metric itself could be useful for data pre-processing (e.g. for establishing an average distance of variable/non-variable stars as a one of extracted attributes).

2.1.1.2 Back of SFA symbols (BOSS)

Introduced in 2015 by Patrick Schäfer [10]. The algorithm is based on a comparison of patterns extracted from time series and contains 3 steps:

1. separation of time series to blocks of same length (sliding windows),
2. transformation of each block using Symbolic Fourier Transformation (SFA),
    (a) a discrete Fourier transform is applied to each block,
    (b) then a symbolic representation of each block is created (SFA word),
3. a construction of BOSS histogram for identifying structural similarities in time series.

BOSS algorithm includes a noise reduction. It is also fast and successful in classification. The author claims that it is able to supersede the best classification algorithms of the time

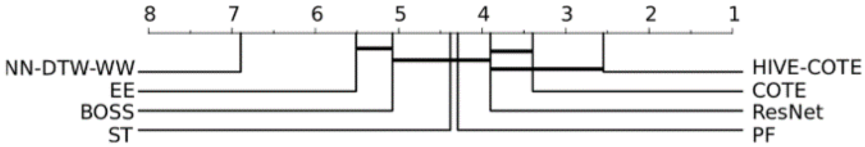


Figure 2.1: Critical difference diagram comparing 8 classifiers over UCR and UEA archives. Source: [8].

and that one of the modifications could reach the top results on UCR data. But in experiments in [7] it performed a little bit worse.

Transformation of time series could be useful for us as well as a part of data pre-processing. We could also use SFA words themselves or BOSS histograms as inputs for artificial neural network.

2.1.1.3 Collective of Transform-based Ensembles (COTE)

First introduced in [13] where authors designed a method that collects several time series classifiers in various domains, related transformations to these domains and metrics that evaluates individual classifiers outputs.

Authors in [13] claims that COTE reaches significantly higher accuracy than other known algorithms (those based on artificial neural networks were not covered). This statement can be supported by [7]. COTE was also a part of experiments in [8] where it was compared with 7 other algorithms. Results are depicted on Fig. 2.1 and COTE took second place with statistically insignificant difference from the winner.

2.1.2 Methods With use of Artificial Neural Networks

Artificial Neural Networks are not new in the area of astroinformatics. They are used in various areas ranging from time series classification (described later in this section) to e.g. automated spectral analysis of large archives such as LAMOST using deep learning [14].

Methods that utilizes artificial neural networks were the main aim of this work. We bring a brief overview of those that we considered as they are usually performing best for time series from various areas. In [12] authors also proved that artificial neural networks can supersede other time series classification algorithms. We can name for example Convolutional network [15], Fully Convolutional Network [5] [12], Residual Network [5], Multi-scalable Convolutional Network [16] [12] or Long Short-term Memory (Recurrent) Network [5] [12].

2.1.2.1 Multi-layer Perceptron

Consists of interconnected layers containing artificial neurons. Neurons in two layers are connected by weighted connection. There is no connection between neurons of the same layer. The output of each neuron consists of a sum of bias and weighted inputs processed by activation function. It is trained by a traditional backpropagation algorithm. More detailed description of MLP can be found in e.g. [17].

In [16] authors attempts to find how each training parameter affects the classifier performance especially in variable stars classification. According to them, the training speed has no significant affect on the accuracy, but size of internal layers has. They also experimented with layers from 4 to 18 neurons and it turned out that those with 4 and 8 neurons performed best. But the main aim of their work was to compare MLP, KNN, SVM and RF in a field of variable stars classification. They attempted to establish a specific classifier for each class and the final class was assigned as a combination of all classifiers outputs. From these, MLP turned out to be somewhere in the middle.

In [7] authors compares algorithms for time series classification that emerged after 2010. It also contains MLP and Naive Bayes (NB), logistic regression, SVM with linear (SVML), quadratic kernel (SVMQ), random forest (RandF) and rotation forest (RotF), 1-nearest neighbor with Euclidean distance (ED) and WEKA C4.5 (C45). From these, MLP achieved better results than DTW, RotF and RandF while it achieved worse results than BN, NB, C45 and logistic regression.

In [18] authors compared several algorithms on the original Kepler data pre-processed by feature extraction and concluded that in terms of accuracy, artificial neural networks performs best (followed by Decision Trees).

2.1.2.2 Convolutional Network

Although first convolutional networks were designed for image recognition [19] [20] [21], today they are commonly used in other domains such as speech recognition and processing [22] [23] or time series analysis [24].

A typical convolutional network consists of convolutional, pooling and fully-connected layers (as depicted on Fig. 2.2). A convolutional layer consists of several convolutional filters attempting to detect patterns (e.g. shapes, gradients or even more complex structures like hairs in image processing domain). The filter is implemented as a matrix used for multiplication of input image. Outputs of convolutional layer are then accumulated in pooling layer made for simplification of inputs their dimensionality reduction. In the end of the network there is a fully-connected network similar to MLP.

In time series domain the convolution can take the place in application of its filter on time series areas. The difference from image processing is that it is applied to one-dimensional inputs. This operation can also be understood as a non-linear transformation of the time series. For example in case of convolution of time series by filter $[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}]$ we talk about

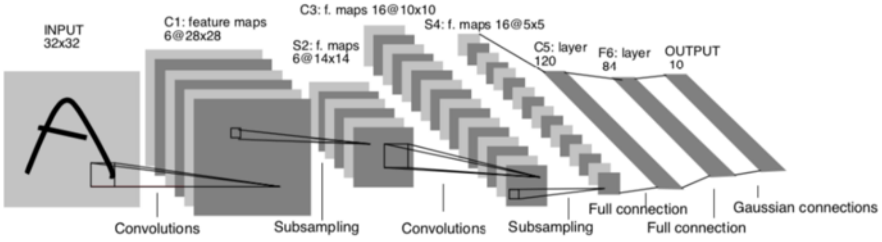


Figure 2.2: Convolutional neural network architecture. Source: [20].

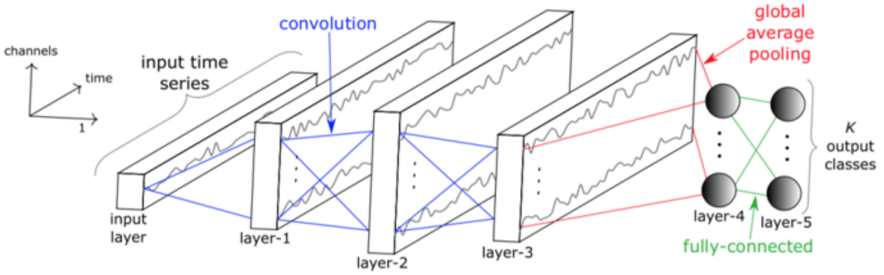


Figure 2.3: Convolutional neural network architecture for time series classification. Source: [8].

moving average with window of 4. A visualization of convolutional network for time series classification is depicted on Fig. 2.3.

In case of traditional algorithms it is usually necessary to take care for intense data preprocessing [15]. The great advantage of convolutional network is that it requires only minimal modifications of input data for the classifier. This is due to convolutional filters that do the preprocessing on their own. According to [8] the CNN is the most used type of artificial neural network for time series classification, most probably for its robustness a relatively good training speed in comparison to MLP and RNN.

2.1.2.3 Fully Convolutional Network

The architecture of Fully Convolutional Network (FCN) is similar to CNN. The only difference is that in the last layer, there is (instead of fully connected layer) Global Average Pooling (GAP) layer and that it does not contain local pooling layers so the time series length remains the same during the processing by the entire network. Using GAP significantly reduces the number of parameters for the network. GAP performs dimensionality reduction of tensor with dimensions $h \times w \times d$ to $1 \times 1 \times d$ by reducing each $h \times w$ matrix to arithmetic average of all of its values. Please note that while in image recognition we work with $h, w > 1$, in time series domain one of these parameters is equal to 1 prior the reduction.

The advantage of FCN is that it does not require any significant data pre-processing or features extraction [25] and it can even be used as a mean of pre-processing [5].

FCN achieves great results in domain of semantic segmentation of images [25] and also in field of time series classification [5] [26]. The first use of FCN for time series classification is described in [25]. Authors compared FCN, MLP and ResNet with DTW, COTE, BOSS and several other algorithms. They used UCR archive for comparison (with 44 data sets in that time). Data pre-processing contained only z-normalization and for FCN and ResNet training the Adam Optimizer was used. Four metrics were used to evaluate results: arithmetic average of errors across all data sets, geometric average of errors across all data sets, number of data sets where the given algorithm performed with lowest error and *Mean Per-Class Error (MPCE)*:

$$MPCE = \frac{1}{K} \sum PCE_k \tag{2.1}$$

where $PCE_k = \frac{e_k}{c_k}$ where e_k is classification error on kth data set, c_k is number of classes in kth data set and K is number of data sets. FCN performed best using these metrics (in 18 of 44 data sets in reached lowest classification error).

Authors in [8] compared current time series classification algorithms. They experimented with 5-layered FCN with ADAM optimizer with learning factor 0.001 and Entropy cost function. Details of parameters and results can be found in [8], but we can conclude that FCN performs worse that ResNet and better than other 7 classifiers.

FCN was also included in experiments in [27] although this work mostly aims to recurrent neural networks in time series classification domain. FCN with MLP were included into these experiments just to compare the accuracy. It turned out that FCN performs much better than all recurent networks in these experiments (simple recurent network and LSTM).

2.1.2.4 Residual Network

In general, adding more neurons and layers should increase the approximation accuracy. The problem is that when the network architecture grows, then the backpropagation algorithm faces the vanishing (or exploding) gradient issues. This problem was identified in [28]. Removing this weakness was the major aim of Residual network (ResNet) in order to allow effective network training for deep learning [26]. Another great advantage is (similarly to CNN and FCN) the need of very little data pre-processing [5].

ResNet performs very well in image recognition domain. In [26] authors designed a contest winning network that on ImageNet data set [29] reached only 3.5%. It also turned out that ResNet can be successfully applied to time series classification. In [25] authors experimented with various types of networks and data sets and ResNet achieved lowest classification error in 8 of 44 data sets. In arithmetic average it took 4^{th} place and in geometric average it took 5^{th} place (of 11 algorithms).

2.1.2.5 Elman‘s Network

Elman’s network is one of the simplest recurrent neural networks (RNN – they store an information about their previous activations in internal memory so the information can spread also among neurons in the same layer; successfully used for sequence data processing [12]). The main difference between MLP and Elman’s network is that Elman’s network is enhanced with context layer that stores an information about activation of neurons from the previous iteration. These activations then affect the output of the layer using recurrent links. These links exist only between ith neuron of context layer and ith neuron of internal layer and have weight equal to 1. The network is trained using real-time recurrent learning method [30].

In [27] authors attempts to use recurrent neural networks in time series classification domain. They compare different neural networks using *UCR Time Series Archive* [6]: MLP, FCN, LSTM and several other types of recurrent networks. They divided recurrent networks into 2 groups: those with dense layer on the output and those with recurrent layer. Their results for recurrent networks are not very conclusive because they did not reach significantly higher accuracy than random classifier. They conclude that according to Wilcoxon signed-rank test:

- replacing recurrent layer by LSTM layer leads to better accuracy,
- adding third layer leads to no significant improvement of accuracy,
- replacing dense layer by recurrent layer has no significant impact to the accuracy,
- increasing number of neurons in one-dimensional recurrent network from 128 to 256 leads to worse classification accuracy.

2.1.2.6 Long Short-term Memory

The main motivation to create LSTM [31] was to be able effectively model dependencies in large time series and also to eliminate problems with vanishing and exploding gradients [5]. The strength of LSTM comes from regulation of spreading of the activation by gates. These gates regulate input, output and internal state of the LSTM cell. Each LSTM cell consists of 3 gates: input, forget and output. The forget gate decides which information will be removed from internal memory. Input gate filters cell inputs and output gate decides what input values and values from internal layer will be sent to the output of the cell.

LSTM was also a subject of experiments in [27] on *UCR Time Series Archive* [6] where is achieves better accuracy than standard RNN, but worse than FCN. Authors also conclude that increasing the number of neurons in layer from 128 to 256 had no significant affect on classification accuracy. Authors in [32] attempted to classify astronomical time series using LSTM and other methods. Their experiments were evaluated using *Balanced Accuracy*:

$$\frac{\frac{TP}{P} + \frac{TN}{N}}{2} \tag{2.2}$$

Where TP is a count of correctly classified positive samples and TN is a count of correctly classified negative samples. P is a count of positive samples and N a count of negative samples. Authors were surprised that LSTM performed bad in their experiments using this metrics. They conclude that the *Balanced Accuracy* for LSTM was only 52%.

It seems that noisy time series are quite a challenge for LSTM and RNN in general. Possible solution can be inspired by [33], where the authors used a conversion into a symbolic representation with a self-organizing map.

2.1.2.7 LSTM Fully Convolutional Neural Network

Introduced in [5] and designed to extend FCNN by LSTM module. Authors state that their solution significantly improves the performance of FCNN with only a small increase of the model. They also state that their solution requires only minimal data pre-processing. They also conclude that their method super-seeds other state-of-the-art methods.

2.2 Other Related Work

In [34] authors performed multi-class classification (instead of our binary one) for the data from original Kepler mission ("K1") that are very similar to ours. Their original results were similarly poor as ours and among others concludes that LSTM is not suitable for Kepler data. They achieved best results with feature extraction. The results are very similar to those described in [35] (again experimented on the original Kepler data). In [18] authors compared several algorithms on the original Kepler data and reached accuracy very similar to ours on similar Kepler K2 data set (only with different extracted features more suiting to the Kepler data set).

There is also a Kaggle[1] competition aiming on original Kepler data, but although it promises high accuracy, it uses highly unbalanced data set and achieved results are therefore not very informative.

In [36] authors works with a totally different light curves data set, but conclude that feature extraction (in their case a set o 7 statistical markers) is a must-have for astronomical time series. They also work purely with time series without any additional meta-data describing the stars themselves. Based on just these information, they conclude that it is not possible to perform good-performing multi-class classification for most of the variability classes. Using Random Forest, Decision Trees and kNN algorithm they reached up to 70% accuracy which they suspect is caused by an imbalanced data set.

---

[1] Mystery Planet (99.8%, CNN): `https://www.kaggle.com/toregil/mystery-planet-99-8-cnn`

Chapter 3

Data Sets

From publicly available, real-world data sets such as *All Sky Automated Survey for Supernovae (ASAS-SN)*, *MACHO Project*, *Microvariability & Oscillations of Stars (MOST)* we eventually chose *BRITE* (enhanced by variability data from *GCVS* catalogue) and NASA's *Kepler* (both original and *K2* mission) as they provide data sets in a shape suitable for machine learning (ML).

Real light curves are quite challenging for ML classification. They are very often noisy due to various physical reasons or due to contamination by other signals. In [32] authors for example filtered out those light curves with a large contamination from the neighboring stars or those with total measured flux or a flux yield significantly lower than object's total flux. The sampling rate varies and especially in case of e.g. BRITE data set we can see that whole parts of the time series are missing. Such sparseness is suspected to be one of the reasons why classification using ML does not work very well [35]. Another suspect for ML classification poor results is the density of the data [35], which is why we selected in case of K2 data an original pre-processed data set with 'smoothed' light curves. This data set also removes incorrect values from the light curves caused by instrumentation orientation change performed by on-board thrusters fired during exposure time. This provides more consistent light curves, but on the other hand it brings in another issue which is sparseness. Kepler data are also a bit specific as they contain time series measured in 2 so called cadences where each have a different sampling rate that affects the density of the data. Some authors overcomes this by simply ignoring them [32]. As we can see, there are many challenges, which is why there are very often used additional meta data like those available in K2 FITS files that comes with photometric data and physical properties of each measured star [35] [32]. This - side by side with feature extraction - is the usual ML framework for Kepler data set. In our work, we focused only on raw light curves and labels established by a respected 3rd party without these meta data.

3.1 BRITE

Data from BRITE project - a group of nanosattelites on lower orbit launched in 2013 and 2014. This data set is made of tabular ASCII files containing (among others) Heliocentric Julian Date and Flux (ADU/s). There are 1119 light curves within this data set. According to GVCS catalogue [37], 601 of them are of variable stars, 279 not variable and 239 cannot be identified by cross-matching in GCVS catalogue.

Beside the fact that variability data comes from a 3rd party archive, the greatest disadvantage of this data set is its variability in sampling rate (in some cases, intervals between samples are less than 1ms on one side and approximately 60 days on the other side). Another disadvantage is that number of samples in each light curve varies (some light curves have less than 10 measurements, while others have tens of thousands of measurements). The average length of light curve is 12642 of samples [2].

3.1.1 General Catalogue of Variable Stars (GCVS)

GCVS [37] is a list of variable stars. Its first version contained 10820 stars and was released in 1948. Since then it was updated several times and today in contains 52011 variable stars (version 5.1 released in 2015). This catalogue allows a cross identification of stars based on their IDs in various catalogues.

Since BRITE data set does not come with an information about star variability, we used this catalogue to add an information about variability by cross matching its ID. By this we were able to obtain a label (7 classes of star variability) for 1119 light curves.

3.2 Kepler

Launched in 2009, Kepler was designed to detect transits of Earth-size planets in the "habitable zone" [34] orbiting Sun-like stars with high photometric precision. The Kepler data products are publicly available and quite rich. We focused on light curves containing flux values measured in two so-called cadences: long with 29.4min and short with 58.89s sampling frequency. The Kepler data are divided into 18 quarters - as each quarter of year the instrument needed to be re-aligned to keep the focus on the same area in the Universe. The data are in a form of astronomy-specific FITS format.

Labels were obtained from the *ASAS Catalogue of Variable Stars*[1] as Kepler does not provide them directly. The matching was done using *Two Micron All Sky Survey (2MASS)*[2] catalogue ID available in both catalogues (in case of Kepler we used its *Kepler Input Cata-*

---

[1] ASAS Catalogue of Variable Stars: `http://www.astrouw.edu.pl/asas/?page=acvs`

[2] Two Micron All Sky Survey (2MASS): `https://irsa.ipac.caltech.edu/Missions/2mass.html`

logue[3] providing these metadata). This, on the other hand, also means that labels are less trustworthy. We were also able to match only a handful of data from the original data set up to quarter 16. More specifically, we worked with 1198 samples of various variable stars and 1198 samples of noise (to keep the data set balanced).

3.3 Kepler K2

Continuing NASA's mission to search Earth-like extra-solar planets in our Galaxy. Data from K2 mission that are subject of this work are publicly available [38] [39] and well documented [40] [41]. In this work we are interested in Kepler K2 light curves containing flux of individual objects in time. For these data, Kepler K2 also provides official catalogue of confirmed variable objects that we can utilize. Based on sampling frequency, we distinguish 2 cadency groups: long with 1765.5s (29.4min) and short with 58.89s. On each Thursday, more than 160000 objects with long cadency and 512 objects with short cadency were measured and archived. The minimal length of measurement was 1/4 of year for long cadency and 1 month for short cadency (with exception of Q4 where module 3 objects were lost due to hardware failure). Light curve file is in a form of time series where all undefined values are represented as NaN (not a number). As a result, we can obtain about 40000 light curves with length up to 1300 measurements [2].

As mentioned before, we used original, but corrected K2 data that excludes observations during thruster firings [42]. The resulting data are more smooth and without irrelevant samples (although with some sparseness). The difference between raw and corrected data is depicted on Fig. 3.1.

3.3.1 Similarity with Kepler "K1" Data

Kepler mission "K2" followed the original NASA's Kepler mission with the same purpose after the instrument stabilization failure. For this reason, these data sets are very close to each other in terms of the content. Original Kepler data also contains light curves containing flux values with two cadences: long with 29.4min and short with 58.89s sampling frequency.

Currently, more research seems to be done on original ("K1") Kepler data, but due to the data similarity, we include their results in our work as well. In [32] authors performed multi-class classification but with poor results (balanced accuracy 52%). They confirm that LSTM does not perform very well for Kepler data (they discuss that it was either due to the limited positive sample size within our data or the sparseness and/or noisiness of real data) and they achieved best results with use of significant attributes extraction (with a balanced accuracy 74.7%). This conclusion was basically confirmed by experiments performed in [35]

---

[3] Kepler Input Catalogue: `https://archive.stsci.edu/kepler/kic.html`

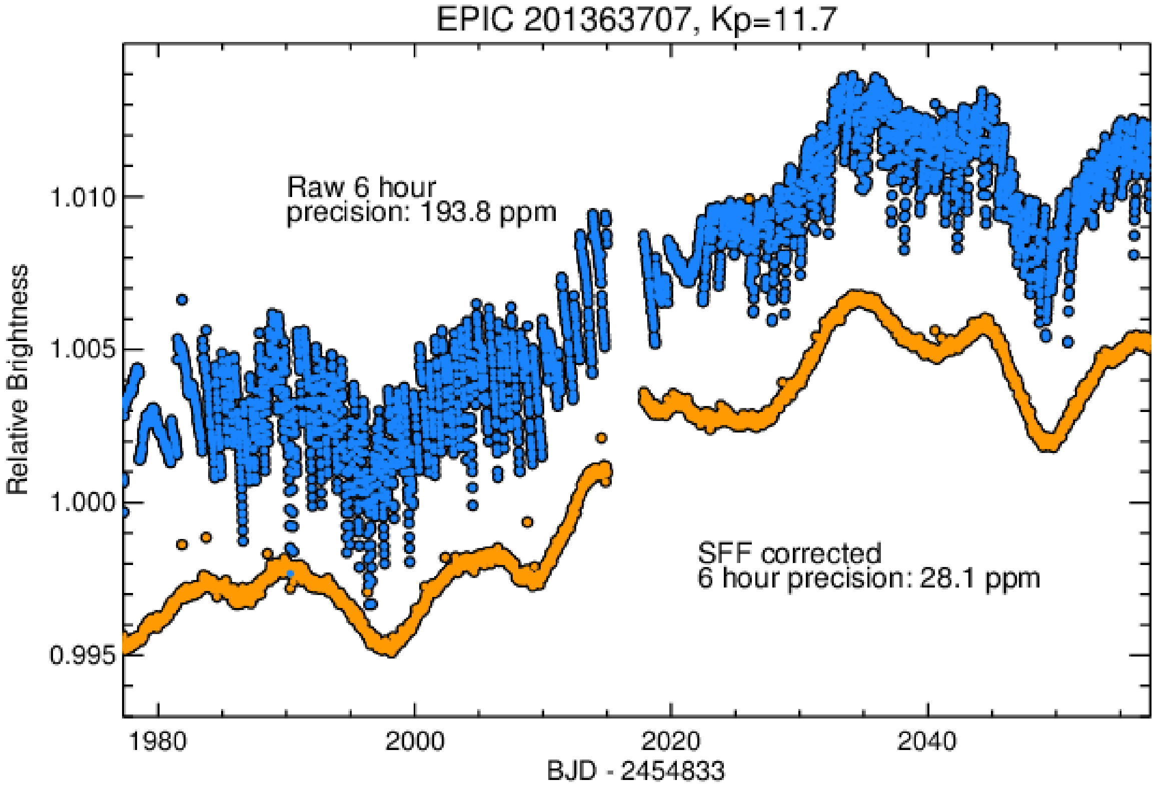


Figure 3.1: An example of the uncorrected fluxes from K2 (blue) and the corrected K2SFF version (orange). Source: [42].

where authors performed also multi-class classification of 150000 objects into 14 variable star classes reaching up to 65-70% accuracy. Authors also mention previous research on Kepler data achieving up to 55% accuracy.

Chapter 4

Artificial Neural Networks Approaches

We decided to compare several methods of time series classification that utilizes artificial neural networks. Our primary goal is to find the best method that will classify time series (light curves) at least binary in a sense of object variability: the light curve contains some period (and is therefore of variable object) or whether there is no period found (an object is not variable) [2] as much as the data will allow us. We started with following approaches:

- multi-layer perceptron classification with own activation function,
- recurrent neural network of type LSTM,
- multi-layer perceptron with own activation function in combination with time series pre-processing using Fourier transformation,
- recurrent neural network of type LSTM in combination with time series pre-processing using Fourier transformation,
- multi-layer perceptron classification with own activation function in combination with some method of significant attribute extraction,
- recurrent neural network of type LSTM in combination with some method of significant attribute extraction (a.k.a. feature extraction),
- convolutional neural network with sigmoid activation function,
- other, not so successful (in terms of our results) artificial neural networks such as fully-convolutional neural networks, MCDCNN and ResNet.

After these we attempted to evolutionary engineer ANN specifically for our binary classification task and establish a framework that can match the classification accuracy of related work.

4.1 Data Pre-processing

We pre-processed both BRITE and Kepler K2 data sets and transformed the raw data (light curves with flux) into a common form digestible by the ANN:

- Balancing data sets so it contains same number of variables and not variables.
- Cutting light curves in order to equal their length.
- Mix the data.
- Generate periodogram.
- Perform Fourier transformation.
- Significant attributes extraction in order to reduce dimensionality of time series data (different techniques).
- Data normalization into interval relevant to selected activation function.
- Splitting the data set to training and test set.

4.1.1 Significant Attributes Extraction and Visualization

During our experiments described later we discovered that both original data sets may not provide clear examples of time series of variable and non-variable stars. This led to a poor accuracy and we were therefore looking for a way how to distinguish these time series by means of significant attributes extraction. We tested the usability of extracted attributes by visualization using Sammon mapping [43]. Sammon mapping attempts to find a low-dimensionality representation of objects in high-dimensional space with as much respect to their original geometric distances as possible. We used it to convert extracted significant attributes to 2D and visualize, hoping to see clear clusters with variable and non-variable time series. Such set of attributes could be then used for further classification using ANN.

Chapter 5

Binary Classification Experiments and Results

As we achieved poor initial results with BRITE data set (as described in 5.1, not all experiments covers it. We were looking at results with 10 following activation functions: exponential, sigmoid, hyperbolic tangent, relu, elu, selu, soft plus, softsign, harp sigmoid, linear. Experiments were performed with artificial neural network containing 3 hidden layers: 16 neurons in input, 34 neurons in first hidden, 16 in second hidden and 64 in third hidden layer [2].

5.1 BRITE Data Set

We had started with a more problematic BRITE data set. We attempted to classify time series by several ways, but eventually with poor results.

5.1.1 Multi-layer Perceptron

The sigmoid activation function was used in the output layer. Training was stopped after 3000 epochs. The learning rate was set to 0.005. For each activation function we trained the MLP 10-times and based on validation data we selected the best model. Its accuracy was then tested on testing data. Results can be seen in Tab. 5.1.

Bad results are probably caused by a small data set. Only 538 light curves came out from pre-processing, these were then divided to a training and test set in 70:30 ratio, 10% of training set was used for validation. For the training, only 338 light curves remained. Another issue was the irregular interval between individual measurements within the time series. Based on these results and results with LSTM, we decided to continue only with Kepler K2 data set.

Act. function	Precision	Recall	Accuracy
Exponential	0	N/A	0.64
Sigmoid	0	N/A	0.64
Hyperb. tan.	0.03	0.18	0.59
Relu	0	0.60	0.64
Elu	0	N/A	0.64
Selu	0	N/A	0.64
Soft plus	0	N/A	0.64
Soft sign	0	0	0.62
Harp sigmoid	0	N/A	0.64
Linear	0	0	0.61

Table 5.1: Results of MLP classification for BRITE data set.

5.1.2 Long Short-term Memory

In this case, the process was a bit different. We used 900 raw time series (as LSTM is supposed to handle it) cross-matched with GCVS catalogue. We divided them into training and test set in 70:30 ratio. Each time series had up to 66500 measurements ("feature vector"). Shorter time series were padded with -1 to this length and all data normalized. With these settings, we achieved the accuracy 60%.

5.2 Kepler K2 Data Set

After attempts with BRITE data set, we switched to a more promising Kepler K2 data set. Configuration was same as in case of BRITE.

5.2.1 Multi-layer Perceptron

The results with the same configuration as in case of BRITE can be seen in Tab. 5.2. Unfortunately, there is just minimal improvement. The best activation function turned out to be Selu that achieved accuracy 0.66. Recall is also interesting metric because it is not such an issue if some non-variable object is classified as variable but it is important to minimize the number of undetected variables. From this point of view, the Elu function performed best. Accuracy and loss function of best models is depicted on Fig. 5.1 and 5.2.

Then we attempted to improve the accuracy by generating so-called periodograms created by conventional statistical analysis. The classifier then attempted to classify these periodograms instead of light curves. Results can be seen in Tab. 5.3 and shows no significant improvement of accuracy. Nevertheless, hyperbolic tangent performed best. We also attempted to establish some custom activation functions listed in Tab. 5.4 (raw light curves were used). As last experiment with MLP, we attempted to use our own activation functions

Act. function	Precision	Recall	Accuracy
Sigmoid	0.98	0.53	0.56
Hyperbolic tangent	0.72	0.61	0.64
Relu	0.67	0.60	0.61
Elu	0.64	0.63	0.63
Selu	0.78	0.62	0.66
Soft plus	0.82	0.58	0.62
Soft sign	0.64	0.58	0.60
Harp sigmoid	0.98	0.49	0.48
Linear	0.73	0.61	0.63

Table 5.2: Results of MLP classification for Kepler K2 data set.

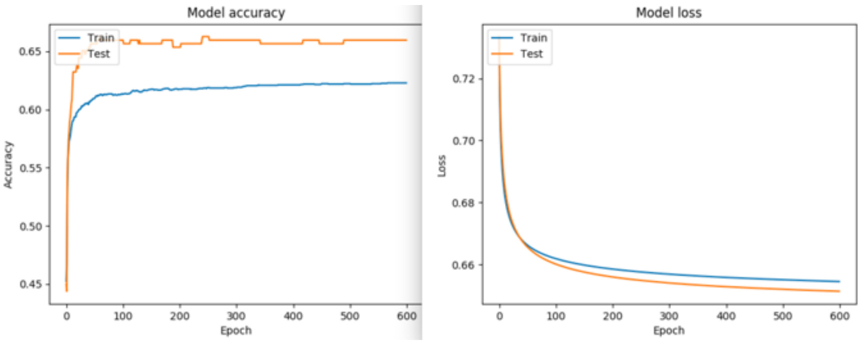


Figure 5.1: Results of MLP training with Elu activation function on K2 data set. Source: [2].

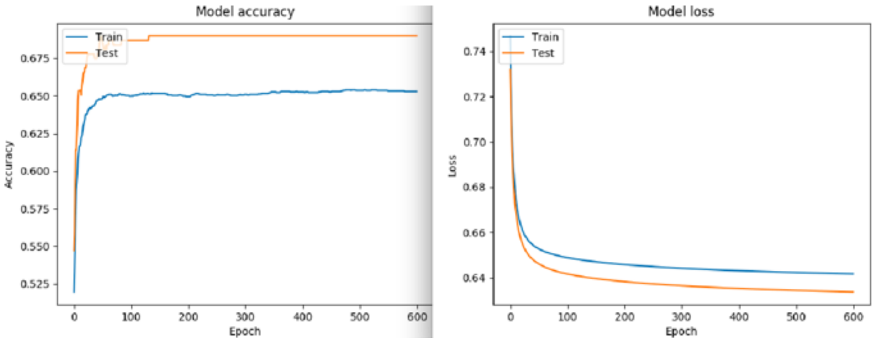


Figure 5.2: Results of MLP training with Selu activation function on K2 data set. Source: [2].

Act. function	Precision	Recall	Accuracy
Sigmoid	1.00	0.49	0.49
Hyperbolic tangent	0.51	0.71	0.66
Relu	0.54	0.70	0.65
Elu	0.63	0.65	0.65
Selu	0.65	0.61	0.62
Soft plus	0.80	0.61	0.65
Soft sign	0.56	0.67	0.64
Harp sigmoid	0.00	0.00	0.50
Linear	0.61	0.65	0.64

Table 5.3: MLP classification for K2 data converted to periodograms.

Act. function	Precision	Recall	Accuracy
tanh(0.1*x)	0	0	0.45
tanh(0.3*x)	0.05	0.53	0.45
tanh(0.5*x)	0.52	0.66	0.59
tanh(x)	0.72	0.68	0.66
tanh(1.5*x)	0.68	0.68	0.65
tanh(2*x)	0.71	0.67	0.65

Table 5.4: MLP classification for K2 data using own act. functions.

with Kepler K2 light curves processed by Fourier transformation. Results are listed in Tab. 5.5. We achieved similar results with Cosine transformation.

5.2.2 Convolutional Network

We decided to compare CNN with MLP. All data from K2 data set were normalized by min-max normalization, we run 5000 epochs with learning rate 0.005 and following network configuration: two convolutional layers with sigmoid act. function, window size of 7 and 6 (or 12) filters, each followed by pooling layer and with output layer with sigmoid activation function. Results are in Tab. 5.6 and on Fig. 5.3.

Act. function	Precision	Recall	Accuracy
tanh(1.1*x)	0.68	0.64	0.64
tanh(1.4*x)	0.75	0.65	0.67
tanh(1.5*x)	0.74	0.65	0.66
tanh(1.7*x)	0.68	0.61	0.61
tanh(2*x)	0.72	0.63	0.63

Table 5.5: MLP classification for K2, own act. functions, FT.

Light curves	Length	Precision	Acc.	Recall
500	800	0.65	0.65	0.65
7502	1300	0.65	0.64	0.64
500	400	0.64	0.62	0.63

Table 5.6: Results of CNN classification for Kepler K2 data with different count of light curves and measurements in each time series (cut to this length).

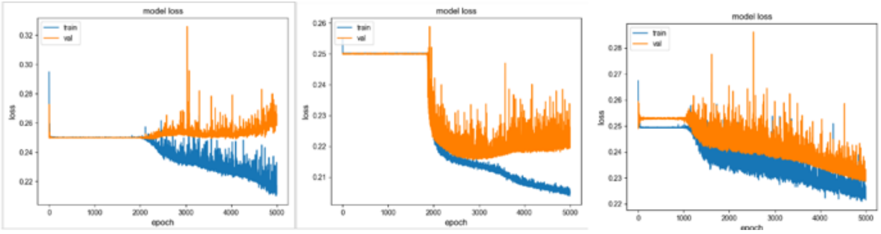


Figure 5.3: CNN experiment loss function chart during training phase (from left: eperiment #1, #2, #3) [2].

5.2.3 Other Artifical Neural Networks

We have been experimenting with several other ANNs including ResNet, Fully-convolutional network, MCDCNN and other configurations of MLP and CNN with even less success than was described above. Their results are therefore omitted from this paper.

5.2.4 Other Significant Attributes Extraction Methods

As mentioned before, we attempted to extract most significant attributes from K2 data set in order to distinguish variable and non-variable objects. To verify this, Sammon projection was used (see Fig. 5.4 and 5.5). The experiment confirmed that the original data does not contain clusters and we need to focus on domain-specific details of the data.

5.2.5 MLP: Improvement of Accuracy

We eventually focused on the most-promising MLP with use of feature extraction and certain domain knowledge. We used Kepler K2 variability metadata that also provides a confidence level of each class label (in a form of probabilistic distribution over all possible classes). The histogram of such label confidence is depicted on Fig. 5.6. For the training purposes, we fine-selected only those time series that has confidence level over 80% and thus are more "clean" for the feature extraction (in other cases there is a significant probability of a bad label) – an experimentally established set of statistical markers (calculated using *tsfresh*[1] framework):

---

[1] tsfresh: `https://tsfresh.readthedocs.io`

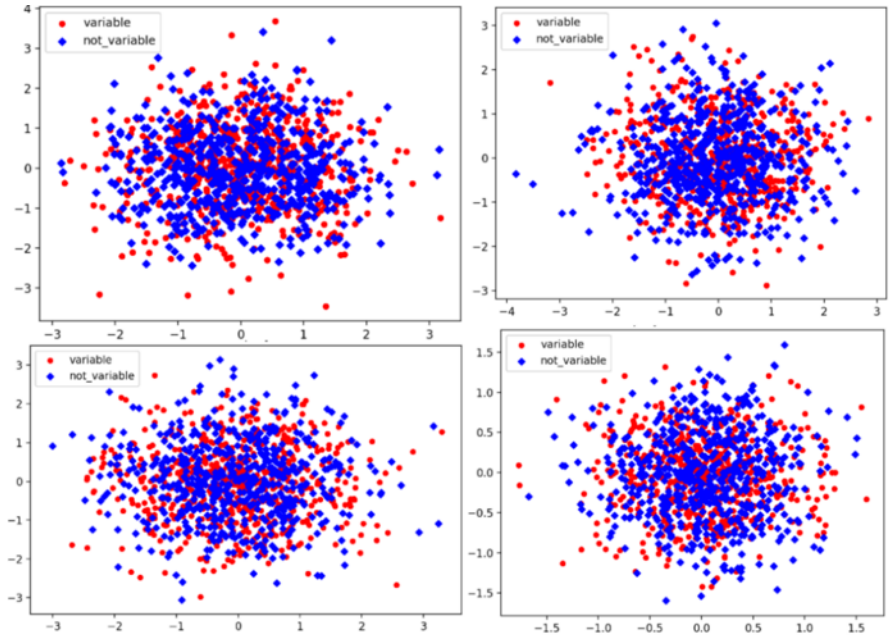


Figure 5.4: Sammon projection, random init.: 500 epochs, different time series length (1200-1225 measurements), extracted attributes or just FT or min-max norm [2].

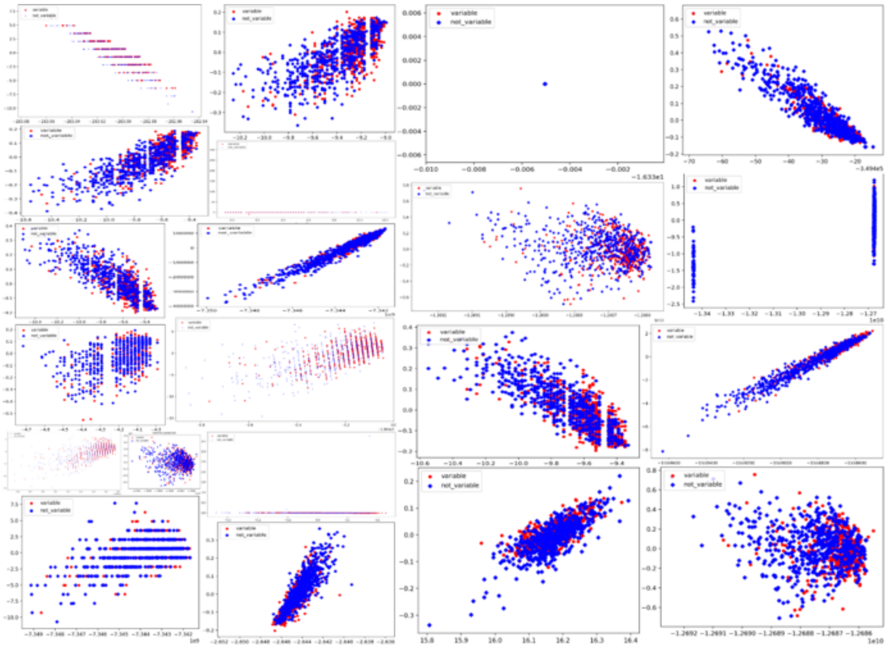


Figure 5.5: Sammon mapping: 500 epochs, time series length: 1200-1225. Attr.: stand. dev., variance, min, max, mean, sum val., median, abs. sum of changes, agg. autocorr., arithmetic coeff., binned entropy, energy ratio, agg. FFT val., first loc. of min/max, mult. max values, index mass quant., linear trend etc. Or processed by FT or min-max normalization [2].

- *Absolute sum of changes* of the time series x: $\sum_{i=1}^{n-1} |x_{i+1} - x_i|$.
- *Aggregated auto-correlation*: $f_{agg} = (R(1), ..., R(m))$ for $m = max(n, maxlag)$ where n is the length of the time series X, $maxlag$ is the maximal number of lags to consider, f_{agg} is mean, variance and standard deviation in our case and $R(l)$ is the autocorrelation for lag l: $R(l) = \frac{1}{(n-l)\sigma^2} \sum_{t=1}^{n-l} (X_t - \mu)(X_{t+l} - \mu)$ with σ^2 being variance and μ mean.
- *Change quantiles* with lower quantile being 0.5, higher quantile being 0.7, using absolute differences and variance as the aggregation function applied to a corridor established by quantiles.
- *CID* - an efficient complexity-invariant distance for time series x attempting to estimate its complexity specified by more peaks, valleys etc. [44]: $\sqrt{\sum_{i=0}^{n-2lag} (x_i - x_{i+1})^2}$.
- *Count above/below mean* returning the number of values in time series x that are above/-below its mean.
- *Energy ratio by chunks* - sum of squares of chunk i out of N chunks expressed as a ratio with the sum of squares over the whole series (we used 10 chunks).
- *Fast Fourier coefficients* $A_k = \sum_{m=0}^{n-1} a_m e^{-2\pi i \frac{mk}{n}}$ for $k = 0, ..., n-1$ where A_k is the kth Fourier (complex) coefficient, n is the length of the time series and a_m is the mth value of time series. We used first 3 Fourier coefficients: their real, imaginary, absolute and angle value.
- *Aggregated Fast Fourier Transformation* - spectral centroid (mean), variance, skew, and kurtosis of the absolute Fourier transform spectrum.
- *C3* measuring non-linearity in time series x [45] by computing $\frac{1}{n-2lag} \sum_{i=0}^{n-2lag} x_{i+2lag}^2 \cdot x_{i+lag} \cdot x_i$ where we used $lag = 350$.
- *Mean value of a central approximation of the second derivative*: $\frac{1}{n} \sum_{i=1}^{n-1} \frac{1}{2}(x_{i+2} - 2x_{i+1} + x_i)$ where n is the length of time series x.
- *Partial autocorrelation* at lag $k = 2$ of time series x [46].
- *Quantile* $q = 0.5$.
- *Range count* of observed values within the interval $[0, 100)$.
- *Ratio beyond* $r\sigma$ - ratio of values that are more than $r \cdot \sigma(x)$ away from the mean of time series x where σ is standard deviation a $r = 100$ in our case.
- *Ratio of count of unique values to count of all values* of the give time series.

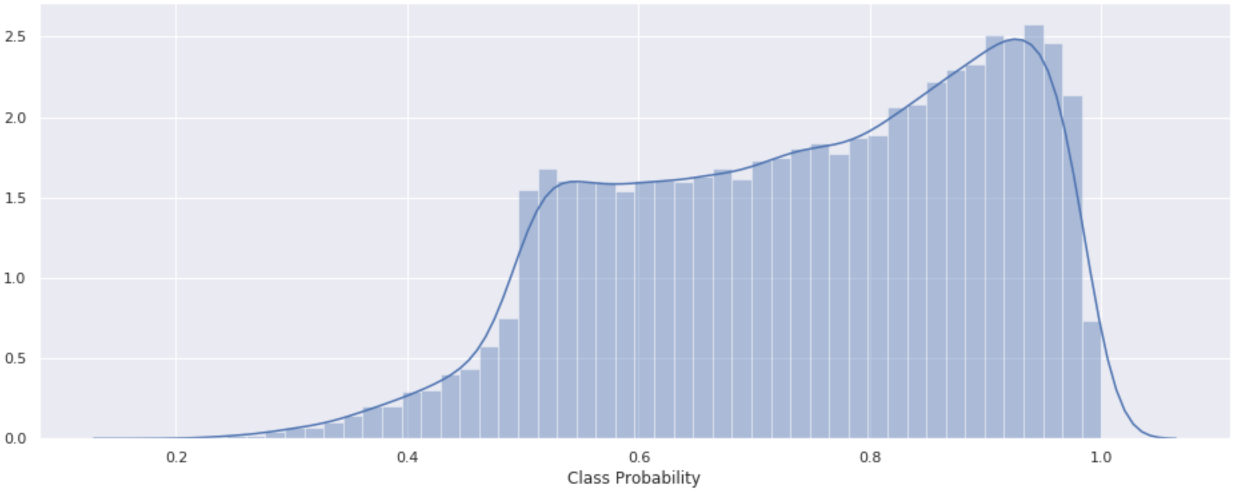


Figure 5.6: Histogram of class probabilities (label confidence) for Kepler K2 data.

- *Skewness.*
- *Power spectrum of the different frequencies* of the given time series.
- *Standard deviation.*
- *Variance.*

On Fig. 5.7 we depict our pre-processed data set. Visualization in 3D was done using non-linear projection called t-SNE [47]. We can see that variable and non-variable stars are now much more distinguishable. In the bottom part where most of non-variable stars are located, we can see a significant number variable stars as well which suggests that there will be a need of rather higher number of layers and neurons in them in order to “bend” the space around them and separate them in high-dimensional space.

We also introduced batch normalization [48] for each hidden layer. The best experimentally found performing network contained 3 dense layers (64, 128 and 256 neurons) and reached accuracy over 70%. To push the accuracy even further, we introduced an evolutionary algorithm called Particle Swarm Optimization (PSO) [49] - a traditional optimization algorithm that is able to cover the whole space of possible solutions on its random agents initialization - to optimize the MLP hyper-parameters by minimization of the classification error (1 - *accuracy*). We optimized the following hyper-parameters:

- Decimal places of extracted statistical markers
    - Encoded as rounded integer value.
- Normalization type (min/max, mean)

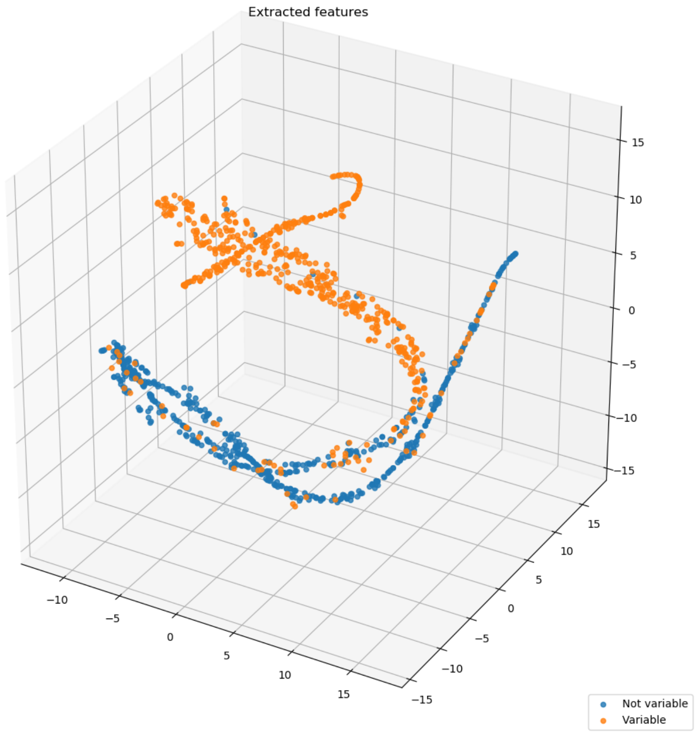


Figure 5.7: Best describing extracted features for most reliable (in a sense of class confidence) time series.

  - Encoded as rounded value of 0 or 1.

- Learning rate
  - From the interval of 0 to 1.
- MLP network structure (number of hidden layers and neurons).
  - Up to 10 hidden layers having different combinations of 8, 16, 32, 64, 128, 256, 512 and 1024 neurons.
  - 43757 possible structures in total.
  - Encoded as rounded integer value that represents one of layers combinations.

The whole process is depicted on Fig. 5.8 and learning took 5 days on modern CUDA-enabled machine (AMD Ryzen 7 2700 with 8 cores / 16 threads, 32GB RAM, nVidia GeForce RTX 2070 8GB, SSD drive). By this, we achieved accuracy 98.55% using following hyper-parameters established by PSO:

- Decimal places of extracted statistical markers: 7 (not so significant).
- Normalization type (min/max, mean): mean.

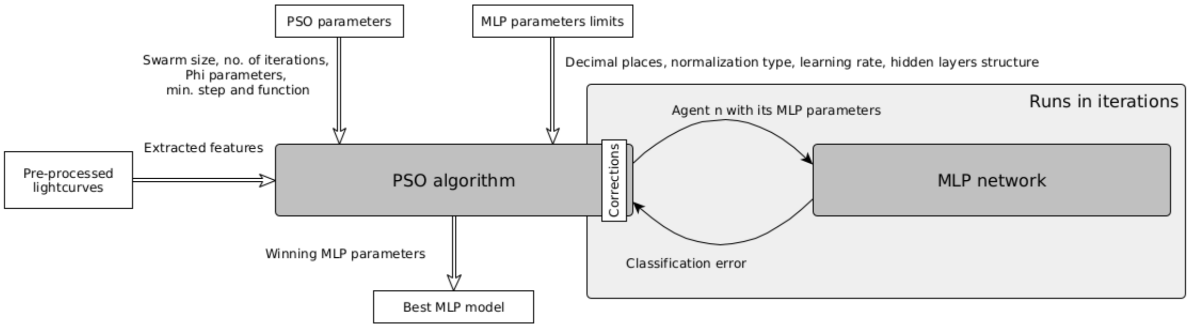


Figure 5.8: PSO over MLP. We start with feeding PSO with initial swarm parameters and limits for MLP parameters that are optimized by PSO. In each iteration of PSO, n agents runs MLP classification with optimized parameters producing the MLP classification error. Based on this error, corrections to MLP parameters of the given agent are made for the next iteration.

- Learning rate: 0.1196596.
- MLP network structure (number of hidden layers and neurons): 8, 16, 16, 32, 32, 32, 1024, 1024, 1024 (this actually corresponds to what we expected based on Fig. 5.7).
- Achieved by 814 light curves in balanced training set.

On Fig. 5.9 we can see the convergence of PSO algorithm. Each point represents a position of agent in 4-dimensional space (each dimension represents one optimized MLP hyper-parameter). The reduction to 2D was done using PCA algorithm. As we can see PSO required just couple of iterations (about 20 or 30) to find the correct area of possible best solutions and then just spent the remaining iterations looking for best performing MLP model within this area. We suspect that further reducing of the area might be difficult due to the stochastic nature of MLP. On Fig. 5.10 depicting the progress of accuracy and hyperparameters of corresponding best fitting MLP models we can see that one of the best fitting models was found within these first tens of PSO iterations. We can also see that high accuracy comes with high count of hidden layers and decimal places while the learning rate remains very low (small changes of internal states of MLP during the training iterations). Each model that reached high accuracy also had similar structure of hidden layers starting from 8 neurons at the beginning layers and ending with 1024 neurons among last layers.

We successfully reproduced the experiment with basically same results and accuracy. It seems that 98.55% accuracy is the current top for our approach to binary classification of Kepler K2 light curves. We are also aware of a bias introduced by optimizing against test accuracy, which is why we watched close the validation accuracy as well as precision and recall where all these metrics were over 90% too. We also tested our approach with data having

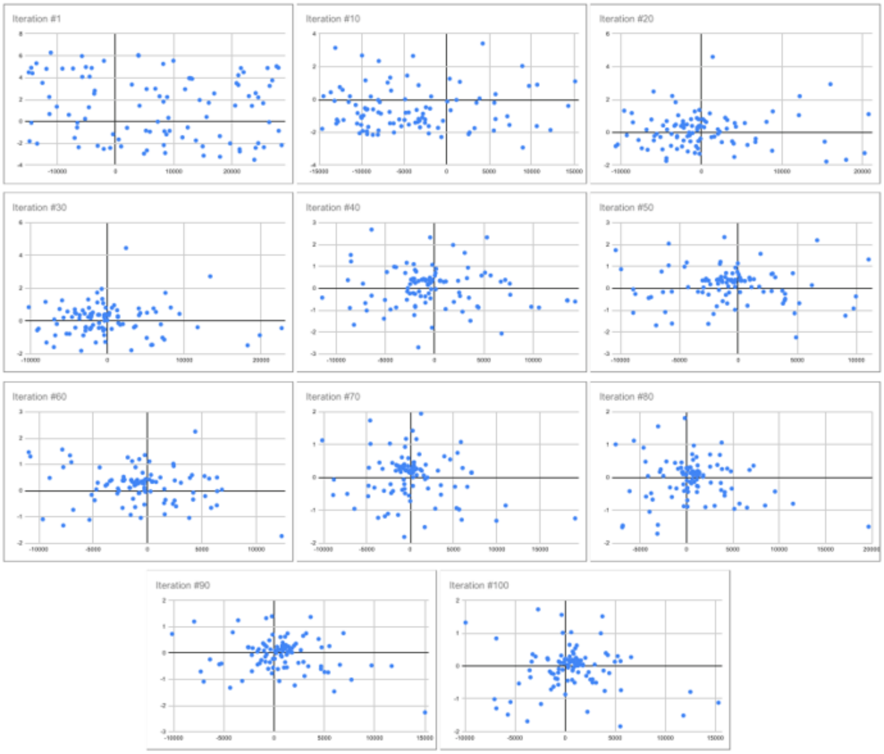


Figure 5.9: PSO agents convergence (after tens of iterations).

		Actual class	
		Variable	**Non-variable**
Predicted class	**Variable**	73	1
	Non-variable	3	61

Table 5.7: Confusion matrix.

label with lower confidence (over 80%) and achieved 82.59% accuracy after only 15 iterations (according to Fig. 5.9 and Fig. 5.10 there is only minimal improvement possible as PSO finds good models very fast). The confusion matrix is depicted in Tab. 5.7.

5.2.6 Results Summary

Similarly to [32] we achieved poor results with LSTM and best results using feature extraction. Although in [32] and [35] authors worked with original Kepler mission data and multi-class classification (instead of our binary one), we were able to reach higher accuracy on a very similar data Kepler K2 data set and binary classification. It seems that the key lies in data pre-processing and selection of correct training data as the confidence level for the label data is often deeply under 50%. Comparison with Kaggle competition results is also problematic

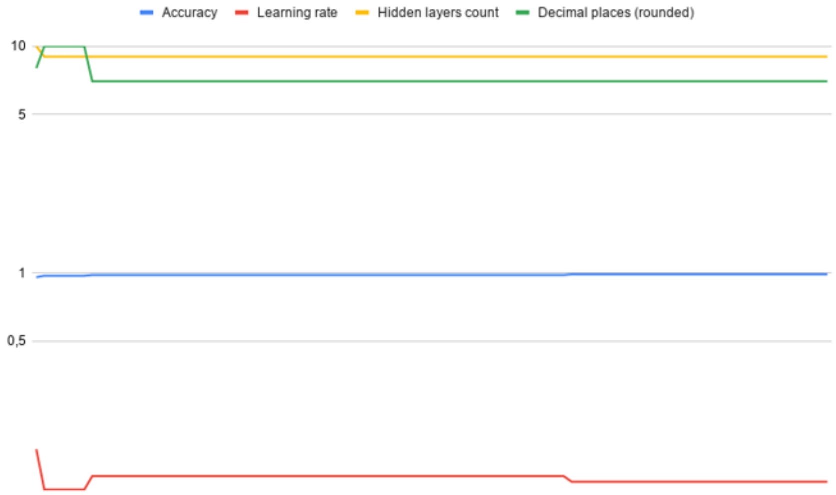


Figure 5.10: Best fitting MLP models over PSO iterations and their hyper-parameters.

not just for the fact that it also uses the original Kepler mission data, but also for using a highly imbalanced data set.

We present another example of "semi-binary" classification experiment in section 6.2.3 where we focused on training a network suitable for distinguishing a specific variability class from each other and from noise.

5.3 Kepler Data Set

We decided to perform some experiments to compare out method with others that focus mostly on original Kepler data. As mentioned in section 3.2, we had 2396 samples available from quarter 1 to 16 which were balanced in a way where 1198 samples were of all variability classes and 1198 were noise. These labels were established by respected 3rd party. For the purpose of the binary classification though, we used only the data from the first four quarters as they are mostly used in related work. This significantly reduced the size of the data set (651 samples together), but the data set was still balanced. The major difference from the K2 data is that the original Kepler data were not "smoothed" by the Kepler team, which turned out to have impact on the classification progress and overall classification accuracy. Giving the size of the data set and our experience with binary classification of Kepler K2 data set, we decided to introduce oversampling (by factor 2) to increase the data set size. We then applied the same algorithm as for Kepler K2 data: extraction of the same features and evolutionary-engineered MLP. In this experiment, the MLP hyperparameters were optimized by approach described later in section 6.2.1. Results are depicted in Tab. 5.8. We let the PSO to perform 58 iterations which is much higher number than in previous experiments. The best achieved

#	DO	BNM	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	Acc
2	0	0	626	487	109	863	836	278	239	608	316	1000	70.45
5	0	0	572	411	809	601	435	220	263	362	325	1000	75.76
10	0	0.03	521	397	107	603	472	199	262	385	314	750	76.52
23	0	0.05	523	399	113	641	547	199	283	397	312	700	78.79
29	0	0.1	610	381	131	655	613	297	301	467	346	778	80.3
30	0	0.08	539	503	118	642	547	243	274	436	309	720	81.06
31	0	0.26	608	470	234	551	664	420	217	502	287	819	81.82

Table 5.8: Binary classification results and best network structures for Kepler data set using evolutionary algorithms. First column (#) indices the iteration number, *DO* stands for Dropout, *BNM* for Batch normalization momentum and *Lx* columns indices the number of perceptrons in each layer. Last columns indices the classification accuracy on a test data set.

accuracy was 81.82% for all label probabilities. We expect that the absence of sophisticated and domain-specific smoothening together with extracted features originally experimentally established for Kepler K2 data set is behind these results. Nevertheless we can again see that PSO converges to the optimal model, just slower. What is also surprising is that batch normalization is almost zeroed in all found best models (together with dropout). We can also notice that deep models with a lot of neurons in hidden layers dominates among the best models.

Chapter 6

Multi-class Classification Experiments and Results

In our binary classification experiments for Kepler K2 there were 6 variability classes with various distribution merged into a single class called ‘variable’. Differences between samples of individual classes could thus make the learning and classification more difficult. We therefore splitted the ‘variable’ class back to 6 individual variability classes with ‘Noise’ as the 7th class and performed the multi-class classification using (primarily) Kepler K2 data set.

6.1 Data Pre-processing

From the current state of the art and our own experiments with binary classification, we know that feature extraction with deep MLP is the most promising approach. We therefore extracted the same statistics as in case of binary classification (see section 5.2.5).

Number of samples in each class is depicted on Fig. 6.1 and we can see that the data set is highly imbalanced. We used a combination of undersampling for large classes (random n samples from the class) and oversampling for small classes (all samples in the class are repeated to collect n samples in total) to get a balanced data set (example is depicted on Fig. 6.2). The exact n number representing the number of samples of each class was also a subject of experiments as it is specific for the data set. We did not want to introduce any unnecessary bias by repeating the same samples for small classes too many times. But we also did not want to miss the opportunity to have enough of different samples for large classes on the other hand. For this reason we evaluated several counts of samples in each class using the best found MLP model from previous binary classification task (see section 5.2.5). To make the classification a bit more difficult, we did not consider the class probabilities ("confidence level") as described in section 5.2.5 and included simply all probabilities. Data were then normalized using min/max normalization and divided in ratio 80:20 to train and test. In Tab.

6.1 we present our encoding table for variability classes. The encoded values are used further in this section in confusion matrices.

Encoded Value	**Class**
0	Noise
1	DSCUT
2	EA
3	EB
4	GDOR
5	OTHPER
6	RRab

Table 6.1: Encoding table for variability classes.

6.2 Experiments

As mentioned before, we started with various counts of samples in each class evaluated using model from section 5.2.5. The counts we tested were 1000 (used as a reference one), 100, 200 and 400 and we attempted to find the least usable data set that provides enough of samples (with respect to large classes) but still does not introduce a bias by too many oversampled small classes. We also experimented with various settings of batch normalization and dropout to see what affect on classification accuracy they have. Results are depicted in Tab. 6.2, 6.3, 6.4 and 6.5. The best preliminary results shows accuracy over 70% using 1000 of samples per class (800 for training; see the confusion matrix on Fig. 6.3) but as mentioned before, these results may be biased by too large oversampling of small classes (e.g. *RRab* that has only 50 unique samples). To rule out this, experiments with only 100 samples per class followed (80 for training) where the overall accuracy was only up to 40%. What is interesting here is the confusion matrix (see Fig. 6.4). It shows that class 6 (which is encoded value of *RRab* with just 50 unique samples) had very little amount of classification errors. This suggests that when we increase the oversampling, we will not introduce such a bias as the samples in this particular class seems to be different enough from other classes. We followed this hypothesis and increased the number of samples for each class to 200 which led to significantly higher classification accuracy (55%). Increased number of samples (even without oversampling) thus leads to better accuracy. From the confusion matrix (Fig. 6.5) we can see that classes 2, 3 and 4 having slightly over 200 unique samples produced a significant amount of classification errors. As there were still some samples left unused in these classes, we decided to use them with an addition of little oversampling and increased the count of samples in each class to 400. This lead to the classification accuracy almost 63% where only the *Noise* class is significantly misclassified (see Fig. 6.6).

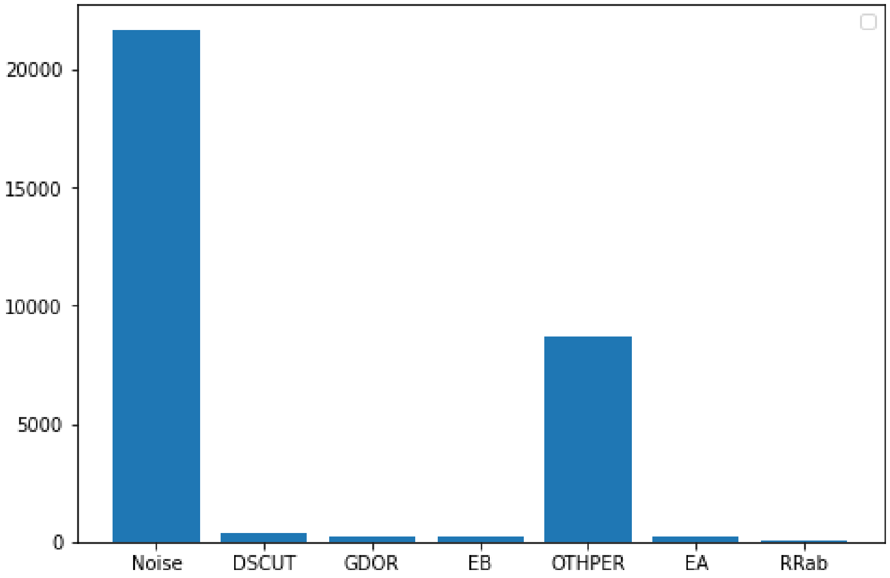


Figure 6.1: Counts of light curves in each variability class in Kepler K2 data set.

We also discovered through naive experiments with the network structure that the best model found for binary classification is not the best fitting for multi-class classification. We were able to reach 47% accuracy on a data set with 100 samples per class, 57% when having 200 samples per class and 67% when having 400 samples per each class. With respect to results in Tab. 6.2, 6.3, 6.4 and 6.5 we can say that increasing the amount of samples leads to better fitting model. To not introduce any unnecessary bias while keeping large-enough data set, we decided to continue with a data set with 400 samples per each class.

6.2.1 Applying the Evolutionary Algorithm

We applied the Particle Swarm Optimization algorithm again. This time (as we performed multi-class classification task and learned new insights from the previous experiments) we optimized a bit different set of parameters (see Fig. 6.7). Each hidden MLP layer can be followed by dropout and batch normalization - with same settings through the whole network. The network itself can have up to 10 hidden layers. So beside global dropout and batch normalization momentum we also optimized the number of perceptrons in each hidden layer (ranging from 0 to 100 where the resulting number was multiplied by factor 10 and rounded so we can have hidden layers with up to 1000 perceptrons - this should be faster for optimization and in practice, tens of neurons makes more difference than individual units). As a result, we optimized 12 parameters this time. Tests were performed on a Kepler K2 data set with 400 samples per class. The PSO was initialized with the same settings as in case of binary classification.

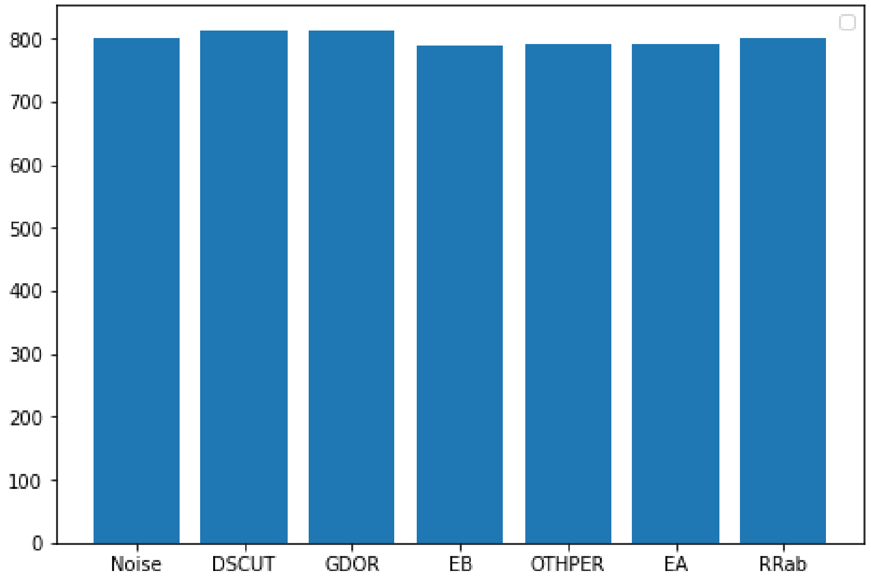


Figure 6.2: Counts of light curves in each variability class in Kepler K2 data set after balancing (to 1000 samples for each class in this example).

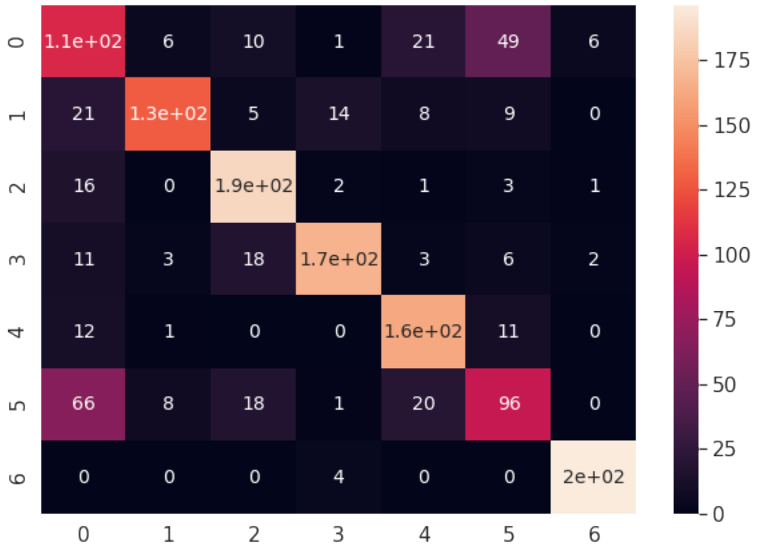
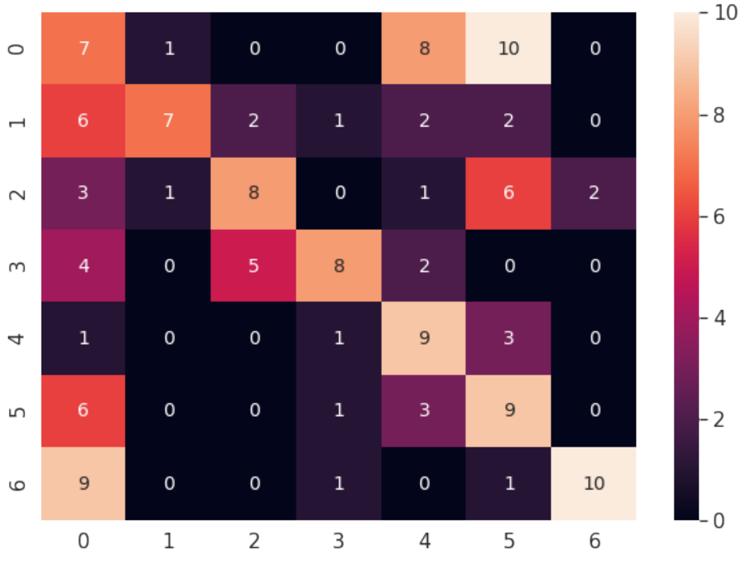
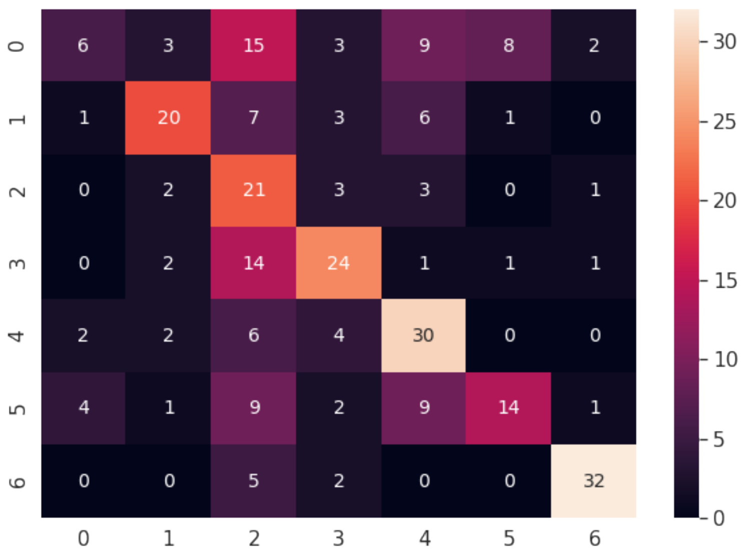


Figure 6.3: Confusion matrix for Kepler K2 multi-class classification with 1000 samples per class. Please see Tab. 6.1 for classes encoding. We can see that OTHPER and Noise are often confused.



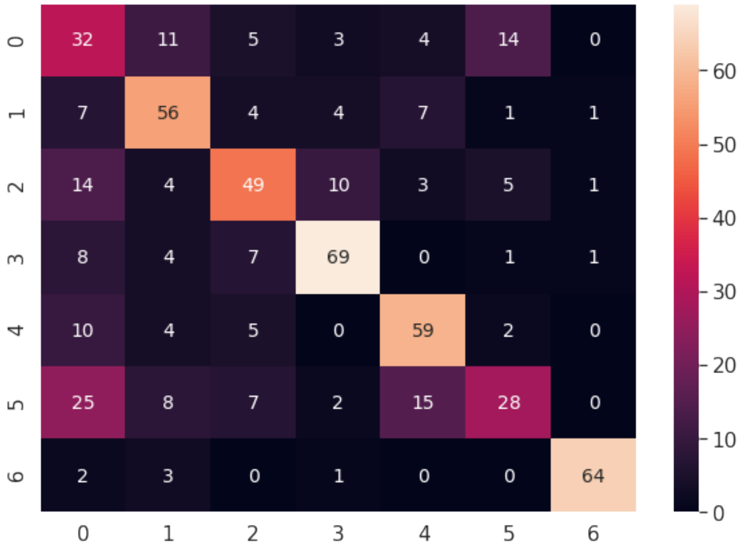
	0	1	2	3	4	5	6
0	7	1	0	0	8	10	0
1	6	7	2	1	2	2	0
2	3	1	8	0	1	6	2
3	4	0	5	8	2	0	0
4	1	0	0	1	9	3	0
5	6	0	0	1	3	9	0
6	9	0	0	1	0	1	10

Figure 6.4: Confusion matrix for Kepler K2 multi-class classification with 100 samples per class. Please see Tab. 6.1 for classes encoding. We can see that *RRab* is (despite to just 50 original samples) correctly classified.



	0	1	2	3	4	5	6
0	6	3	15	3	9	8	2
1	1	20	7	3	6	1	0
2	0	2	21	3	3	0	1
3	0	2	14	24	1	1	1
4	2	2	6	4	30	0	0
5	4	1	9	2	9	14	1
6	0	0	5	2	0	0	32

Figure 6.5: Confusion matrix for Kepler K2 multi-class classification with 200 samples per class. Please see Tab. 6.1 for classes encoding. We can see that *GDOR*, *EA* and *EB* are often misclassified.



	0	1	2	3	4	5	6
0	32	11	5	3	4	14	0
1	7	56	4	4	7	1	1
2	14	4	49	10	3	5	1
3	8	4	7	69	0	1	1
4	10	4	5	0	59	2	0
5	25	8	7	2	15	28	0
6	2	3	0	1	0	0	64

Figure 6.6: Confusion matrix for Kepler K2 multi-class classification with 400 samples per class (no batch normalization nor dropout). Please see Tab. 6.1 for classes encoding. We can see that only *Noise* class is classified significantly incorrectly.

From Fig. 6.8 and Tab. 6.6 we can see that good optimizers are found quite quickly among the first iterations. The best one (with accuracy 73.04% over 7 classes) was found after just 11 iterations which took 4 days. The best fitting model contained 10 hidden layers with dropout and batch normalization layers following them. Details can be found on the last line of Tab. 6.6 where we can see that dropout was ignored while batch normalization momentum was set to 0.44. The model utilized whole 10 possible hidden layers with quite a lot of perceptrons in later layers. We can also notice a small change in best fitting models structure in respect to the binary classification as here (after some initial attempts) there are also usually lower numbers of perceptrons in the initial hidden layers, but this time there are more of them than the size of the input vector. The number of perceptrons then progresses to larger numbers in later layers similarly to the binary classification. What can also be observed is a progression from the initial best fitting models that did not utilize all the possible 10 hidden layers (just most of them) to the final best fitting model that used all 10 hidden layers. Based on this it seems that deep learning with MLP and elaborated feature extraction could be a way how to effectively classify this kind of data - especially when the network structure is engineered by the evolutionary algorithm. From confusion matrix of the best fitting model (Fig. 6.9) we can see that there are two classes that still seems to be tricky for the classifier - this could be addressed by adding more features to the data set (like for example photometric metadata) or further fine-selection of the training data.

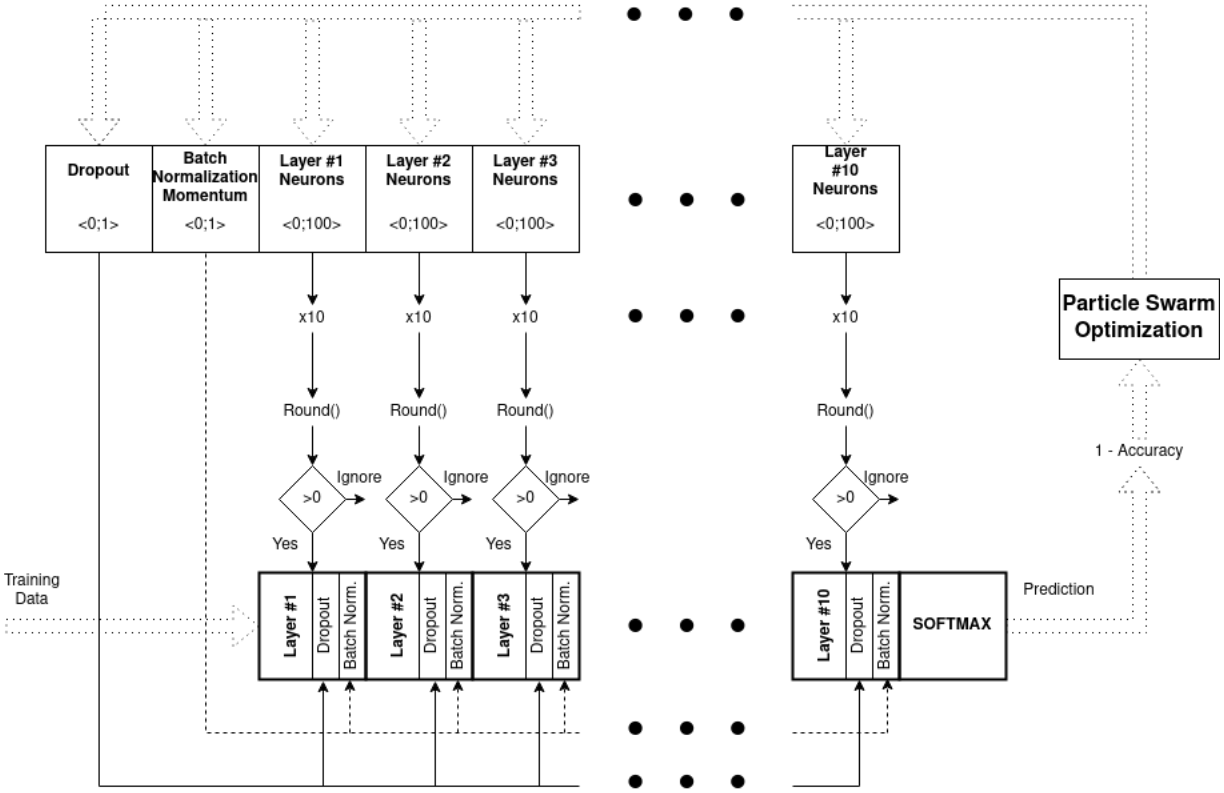


Figure 6.7: Visualization of the cost function for optimizing the Kepler K2 multi-class classifier. Each PSO agent comes with a dropout, batch normalization momentum and number of neurons in MLP layer as input arguments. The MLP arguments are then multiplied by 10 and rounded. If not zeroed, then a layer with corresponding number of neurons is added, followed by dropout and batch normalization set directly. The cost function then performs learning of such MLP and calculates the accuracy, which is subtracted from 1 and serves as the output to be minimized.

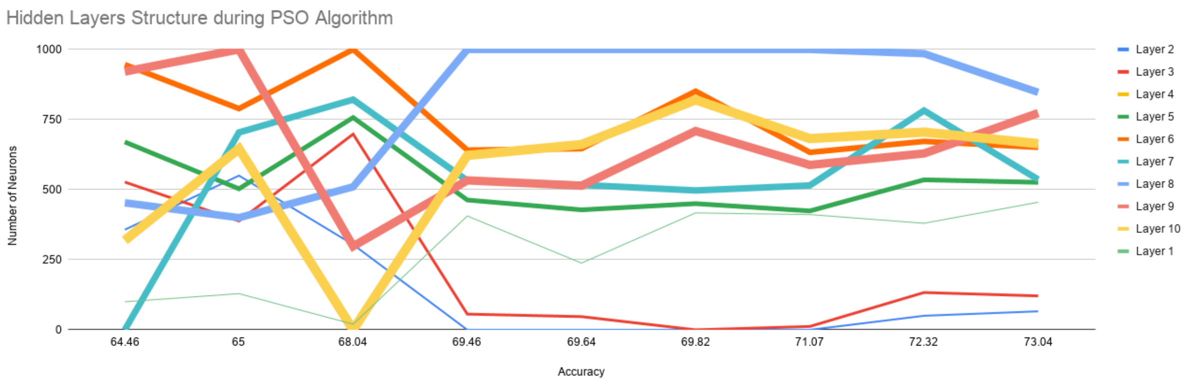
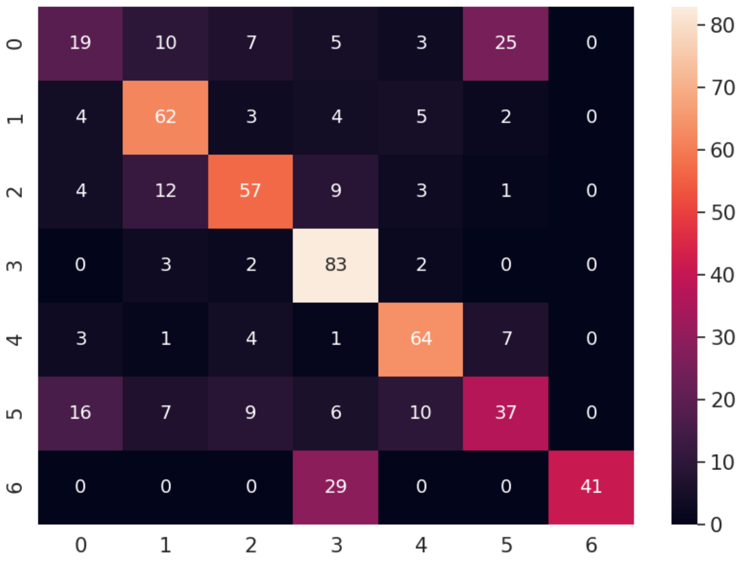


Figure 6.8: Numbers of neurons in each hidden layer over the PSO algorithm run (with accuracy depicted on x axis). Thicker line means later layer.



	0	1	2	3	4	5	6
0	19	10	7	5	3	25	0
1	4	62	3	4	5	2	0
2	4	12	57	9	3	1	0
3	0	3	2	83	2	0	0
4	3	1	4	1	64	7	0
5	16	7	9	6	10	37	0
6	0	0	0	29	0	0	41

Figure 6.9: Confusion matrix for the best fitting multi-class classification MLP model found using PSO.

Batch normalization	**Dropout**	**Iterations before overfitting**	**Average Accuracy**	**Notes**
N/A	N/A	∼300	73.128	Confusion matrix shows that OTHPER and Noise are very similar to each other
N/A	0.2	∼800	71.656	Confusion matrix shows that OTHPER and Noise are very similar to each other
Momentum=0.99	N/A	∼100	52.284	Increased speed, but worse accuracy
Momentum=0.01	N/A	∼120	65.114	Increased speed, but worse accuracy
Momentum=0.3	0.2	∼300	58.584	Middle speed, worse accuracy - both batch normalization and dropout leads to worse accuracy in this case

Table 6.2: Multi-class classification results for Kepler K2 data set with 1000 samples per class.

6.2.2 Extending the Data Set Using Photometric Metadata

We again used our data prepared in section 6.2 joined with the official metadata[1]. It provides multiple additional photometric data where some of them (estimated physical parameters of individual stars) can be particularly useful:

- Effective Temperature (K),
- log Surface Gravity (cgs),
- Metallicity (dex),
- Stellar Radius (solar units),
- Stellar Mass (solar units),
- Stellar Density (solar units),
- Extinction (mag).

These additional metadata are usually tightly related to the star class itself so we expected an increase of the classification accuracy as it was the case in related work working with the data from the original Kepler mission. We simply normalized these data during the preprocessing. They were then attached to the feature vector so it contained the extracted features and newly the photometric metadata as well. From Fig. 6.10 and 6.11 though we

---

[1] Official Kepler K2 metadata: `https://archive.stsci.edu/k2/epic/search.php?action=Search`

Batch normalization	**Dropout**	**Iterations before overfitting**	**Average Accuracy**	**Notes**
N/A	N/A	∼100	30.074	Confusion matrix shows that the smallest (in terms of original samples) class *RRab* is classified usually correctly.
N/A	0.2	∼200	35.286	Dropout increases learning time, but also the accuracy.
Momentum=0.99	N/A	∼85	37.00	Batch normalization with high momentum lowers the learning time and increases the accuracy.
Momentum=0.01	N/A	∼90	39.856	Batch normalization with low momentum lowers the learning rate and increases the accuracy even better than with high momentum.
Momentum=0.3	0.2	∼200	38.714	Combination of batch normalization and dropout corresponds to the other results.

Table 6.3: Multi-class classification results for Kepler K2 data set with 100 samples per class.

can see that we should not expect too much from these data as they are not clearly separable from each other (they do not make any clear clusters).

Our results basically confirms that. Despite having the additional photometric metadata in our feature vector, these data in their raw, just normalized form do not provide any additional information useful for classification. In fact, we saw small decrease in classification accuracy (69.82% - see Tab. 6.7). This is a bit surprising if we consider that in related work, photometric metadata were crucial to achieve similar classification accuracy on a very similar data set. We expect this to be because of its 'naive' use (only normalization was performed), but it also shows that our approach is robust enough to perform competitively even without these metadata, only with light curves themselves. This is actually a good sign because obtaining photometric metadata is error-prone. It requires a cross-matching to some variability catalogue, usually based on coordinates only which may not be precise. Also, photometric metadata are subject of research so they may not be precise neither. Using only light curves data is then more robust.

Learning itself took approximately 3 days on our test machine and best model was found after just 4 iterations. From the best models found over iterations (see Tab. 6.7) we can see that dropout was always zeroed. Later models, on the other hand, benefits from gentle batch normalization. Hidden layers structures shows again that deep networks with high number

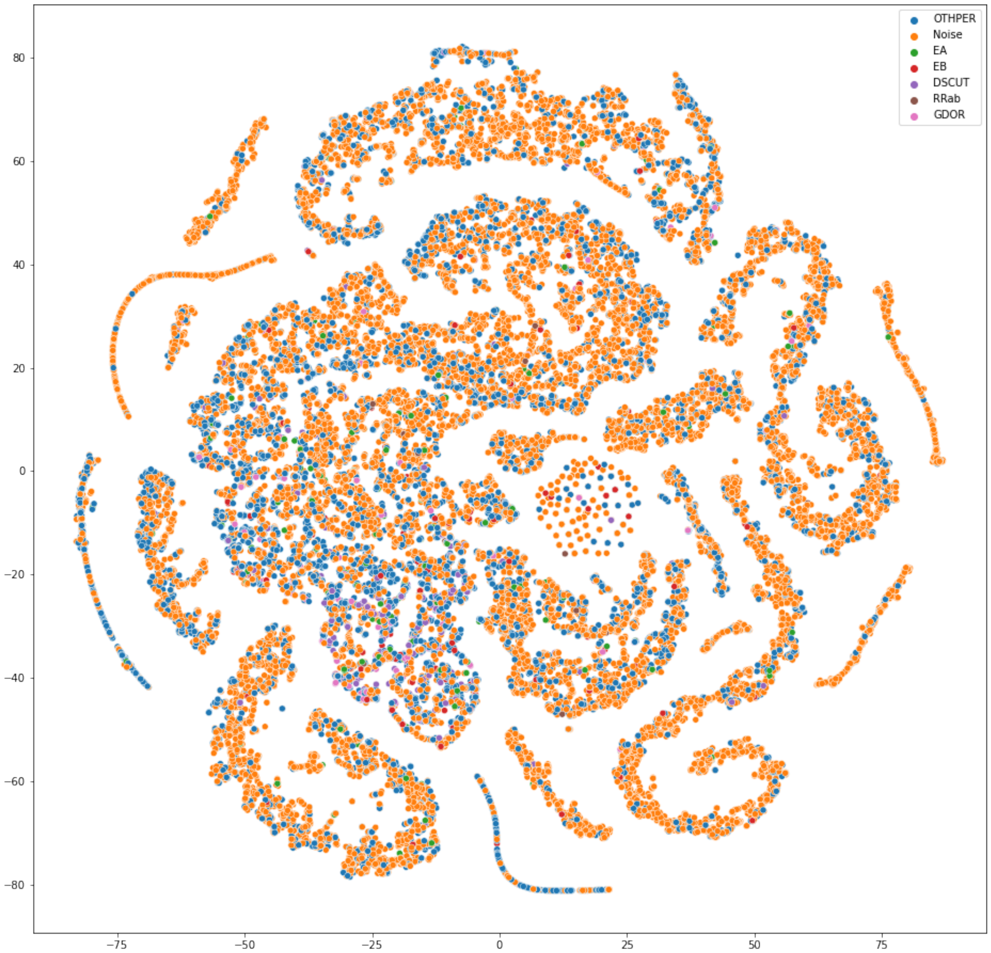


Figure 6.10: Photometric metadata separability (vizualized using t-SNE algorithm). Each point represents normalized photometric metadata of its respected star non-lineary transformed from the original Euclidean space to Euclidean plane.

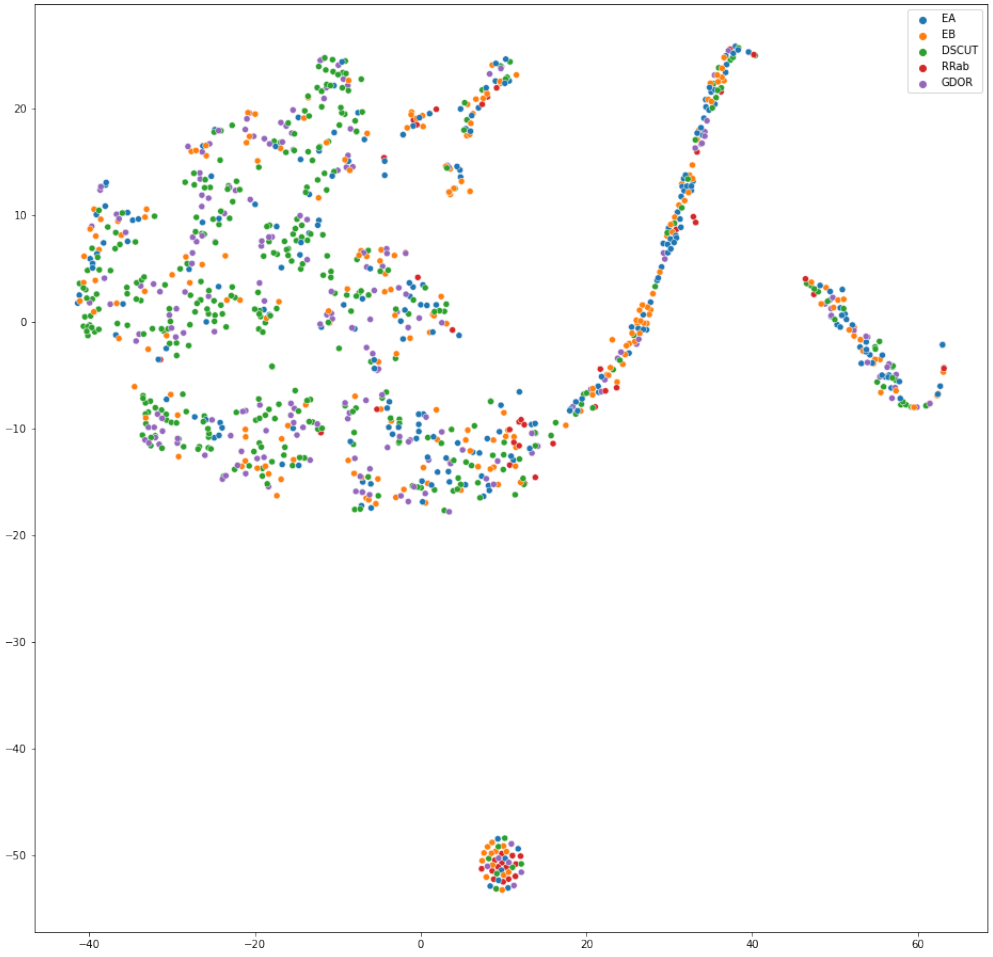


Figure 6.11: Photometric metadata separability (vizualized using t-SNE algorithm). Classes *Noise* and *OTHPER* are omitted in this vizualization as they are too dominant. Each point represents normalized photometric metadata of its respected star non-lineary transformed from the original Euclidean space to Euclidean plane.

Batch normalization	**Dropout**	**Iterations before overfitting**	**Average Accuracy**	**Notes**
N/A	N/A	∼200	51.57	Confusion matrix shows that classes 2, 3 and still produces significant amount of classification errors.
N/A	0.2	∼370	55.356	Dropout increases the learning time, but also the accuracy.
Momentum=0.99	N/A	∼120	48.428	Batch normalization with large momentum decreases the learning time, but also the accuracy.
Momentum=0.01	N/A	∼100	52.786	Even small momentum of batch normalization significantly decreases the learning time while still slightly increases the accuracy.
Momentum=0.3	0.2	∼200	51.358	Combination of dropout and batch normalization seems to have no significant improvement.

Table 6.4: Multi-class classification results for Kepler K2 data set with 200 samples per class.

of neurons in later layers performs best. Confusion matrix (see Fig. 6.12) shows very similar classification errors as in case of previous experiment without metadata (see section 6.2.1).

6.2.3 Training a Fast, One-class Classifier

Looking at the confusion matrices, where usually *Noise* and *OTHPER* classes makes the most classification errors, we were wondering if it is possible to train by our method a classifier specific for each variability class. In this case, we therefore performed a binary classification again. We used our data set prepared in section 6.2 with 400 samples for each class and without photometric metadata. We kept the main classified class (e.g. *GDOR*) and selected the same number of samples from other classes to get approximately 400 balanced samples together as the other class.

Combined results for all variability classes are shown in Tab. 6.8 and ranges from 80% to 97%. From this table, we can see that for example dropout was always zeroed while batch normalization momentum was always very close to the maximum value. This is actually similar to the previous binary classification experiment. It seems that batch normalization is crucial for binary classification - more than for multi-class classification. We can also see that

Batch normalization	**Dropout**	**Iterations before overfitting**	**Average Accuracy**	**Notes**
N/A	N/A	~240	58.964	Confusion matrix shows that only *Noise* class is significantly misclassified.
N/A	0.2	~550	62.608	Dropout significantly increases the learning time, but also the accuracy.
Momentum=0.99	N/A	~100	52.00	Batch normalization with high momentum significantly decreases the learning time, but also the accuracy.
Momentum=0.01	N/A	~150	60.606	Batch normalization with low momentum speeds up the learning as well as slightly increases the accuracy.
Momentum=0.3	0.2	~300	55.466	The combination of batch normalization and dropout performed the worst.

Table 6.5: Multi-class classification results for Kepler K2 data set with 400 samples per class.

even in this case, only deep MLPs performs best. Most of the best models were found after just 4 iterations (about half of the day of training on our test machine), maximum was 12 iterations.

We had two major concerns about these experiments. One was the *OTHPER* class, that is (based on various confusion matrices from previous experiments) very often misclassified as *Noise*, but the actual result (80.75%) while using only a fraction of examples was not as bad as we expected. Second concerns was related to oversampling strategy for small classes, like for example *RRab*, that was highly oversampled and we were afraid of introduced bias here. Its result (96.89%) made us worried a bit, but looking on other classes, like for example *EA* and *EB* with almost same number of samples, but with accuracy different by 7% and especially *DSCUT* class with the least oversampling, yet one of the highest accuracies, we did not see any correlation here.

Creating classifiers for specific classes can be useful for several reasons. We can for example train a classifier that will utilize the outputs of the individual, class-specific classifiers and will learn to perform a "weighted" multi-class classification. Second (specific to Kepler K2 data set) can be the *OTHPER* class. This class is basically a container for other, not specified periods found in the light curve. From the domain point of view, this can imply new, so far unknown phenomena of undiscovered exoplanets or at least to serve as a quick filter to find candidate light curves for further analysis.

#	DO	BNM	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	Acc
1	0.01	0.27	357	527	670	125	946	0	453	921	319	100	64.46
1	0	0.66	550	388	503	0	789	704	400	1000	646	129	65
2	0	0.31	304	698	757	0	1000	821	510	298	0	20	68.04
2	0	0.43	0	56	463	0	641	532	1000	533	622	406	69.46
6	0	0.28	0	47	428	0	646	517	1000	514	661	238	69.64
6	0	0.41	0	0	450	0	851	497	1000	709	820	417	69.82
7	0	0.38	0	12	424	0	633	515	1000	588	682	411	71.07
7	0	0.36	50	133	535	152	672	782	985	629	705	380	72.32
11	0	0.44	66	121	526	66	651	536	847	773	664	455	73.04

Table 6.6: Multi-class classification results and best network structures for Kepler K2 data set with 400 samples per class using evolutionary algorithms. First column (#) indices the iteration number, *DO* stands for Dropout, *BNM* for Batch normalization momentum and *Lx* columns indices the number of perceptrons in each layer. Last columns indices the classification accuracy on a test data set.

#	DO	BNM	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	Acc
1	0	0.86	290	124	279	201	212	268	773	0	1000	597	66.43
2	0	0.21	470	384	747	456	0	253	49	502	942	573	67.14
2	0	0.07	668	0	470	113	834	0	33	138	0	1000	67.86
4	0	0.33	707	135	682	803	888	0	610	20	779	523	69.82

Table 6.7: Multi-class classification results and best network structures for Kepler K2 data set with 400 samples per class and photometric metadata using evolutionary algorithms. First column (#) indices the iteration number, *DO* stands for Dropout, *BNM* for Batch normalization momentum and *Lx* columns indices the number of perceptrons in each layer. Last columns indices the classification accuracy on a test data set.

Class	**DO**	**BNM**	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**	**L7**	**L8**	**L9**	**L10**	**Acc**
GDOR	0	0.99	334	333	68	0	0	437	418	820	511	711	81.99
	0	0.92	280	79	551	573	458	561	549	850	875	391	83.23
	0	0.99	291	74	0	92	83	19	65	450	708	916	84.47
	0	0.99	9	54	451	253	259	511	347	950	905	478	85.09
	0	0.97	148	38	457	476	518	485	539	931	998	635	85.71
DSCUT	0.17	0.99	370	182	0	948	269	728	889	609	223	0	77.02
	0	0.95	173	0	379	245	461	195	264	690	216	321	78.88
	0	0.96	0	827	945	0	266	990	0	0	0	221	86.34
EA	0	0.96	206	686	621	319	678	776	944	1000	727	59	76.4
	0	0.77	762	219	153	0	216	1000	1000	613	0	0	77.64
	0	0.99	734	702	168	0	329	554	939	320	988	570	78.26
	0	0.87	801	320	146	156	270	753	443	544	428	289	80.12
EB	0.07	0.92	919	292	619	534	910	411	537	706	571	0	83.23
	0	0.99	364	0	906	670	1000	292	617	677	398	12	86.96
RRab	0.08	0.99	962	367	183	463	532	514	593	638	241	784	91.93
	0.05	0.99	382	711	234	509	286	310	463	0	848	559	92.55
	0	0.9	270	699	157	553	342	562	960	310	1000	488	93.17
	0.03	0.99	590	549	0	267	224	612	756	0	1000	886	96.89
OTHPER	0.13	0.75	543	0	475	0	552	421	496	130	876	0	79.5
	0.03	0.8	637	0	495	0	557	485	315	309	917	818	80.12
	0.03	0.75	459	0	527	19	533	531	569	291	871	378	80.75

Table 6.8: Results and best network structures of binary classification for each variability class in Kepler K2 data set with 400 samples per class using evolutionary algorithm. *DO* stands for Dropout, *BNM* for Batch normalization momentum and *Lx* columns indices the number of perceptrons in each layer. Last columns indices the classification accuracy on a test data set. Highlighted lines indices the best found model structure for the class.

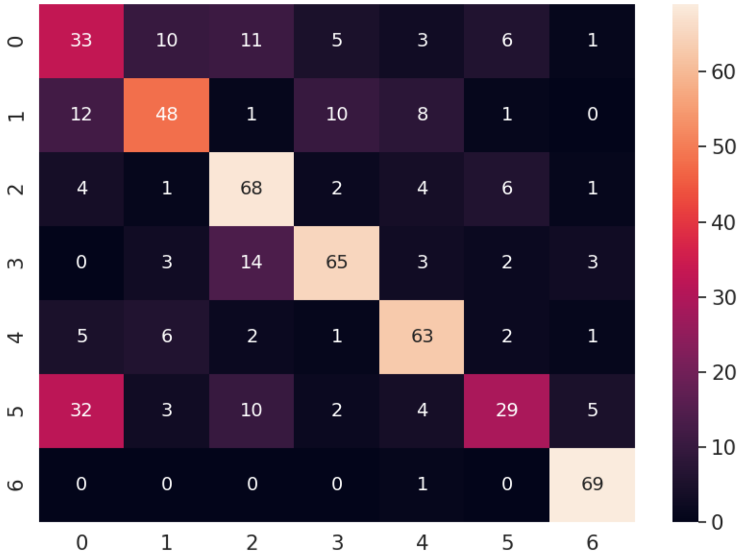


Figure 6.12: Confusion matrix for the best fitting multi-class classification MLP model for data extended with photometric metadata found using PSO.

6.2.4 Testing on Original Kepler Data Set

We tested our approach on the original Kepler data as well, but based on our experience with the binary classification of these data (see section 5.3) we did not expect too much from it as the data itself are not smoothed at all and the number of classes is larger. Also, the extracted features were experimentally established for Kepler K2 data set.

The original data set contained 19 classes, but we dropped out samples for some small classes containing marginary samples at the edge between two classes, which left us 9 classes. We also applied oversampling and undersampling to balance the data to the same number of samples in each class.

Results are depicted in Tab. 6.9. From the learning progress we could also witness issues with the capability to learn itself. Based on our experience with other approaches, we expect that this is mostly caused by too much noise and/or not conclusive extracted features (originally established for smoothed Kepler K2 data). This led to a rather poor accuracy 24.14%. Similarly to the binary classification we can see rather high number of neurons in hidden layers and also the use of all 10 hidden layers which may imply the issues with noise and/or not conclusive extracted features.

#	DO	BNM	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	Acc
1	0	0.46	106	1000	239	1000	1000	1000	210	384	890	12	14.66
3	0.02	0.49	17	799	538	1000	827	985	428	410	802	316	15
5	0	0.28	263	910	414	854	874	939	425	646	970	114	17.24
5	0	0.42	689	841	245	891	648	975	191	636	699	244	24.14

Table 6.9: Multi-class classification results and best network structures for Kepler data set using evolutionary algorithms. First column (#) indices the iteration number, *DO* stands for Dropout, *BNM* for Batch normalization momentum and *Lx* columns indices the number of perceptrons in each layer. Last columns indices the classification accuracy on a test data set.

6.2.4.1 An Idea of Establishing the Best-Fitting Extracted Features

As a further task, we plan to introduce an evolutionary-engineered extracted features. A list of all possible extracted features along with their possible parameters (both numerical and categorical) will need to be encoded to real numbers as subjects of the optimization using the evolutionary algorithm cost function. Couple of approaches are possible here:

1. Create a vector of numbers (each from range $< 0; 1 >$) representing all possible extracted features. The cost function would round these numbers to get 0 or 1 meaning that such feature should be extracted or not. Parameters of the features (e.g. number of FFT coefficients) would be left to default and taken from a map based on an index of the feature in the vector. This is the most direct approach producing a small vector of optimized parameters (leading to a relatively fast optimization), but loosing some information from the data due to just default feature parameters.
2. Extend the previous vector by encoding the feature parameters as well:
   - numerical directly,
   - categorical either as rounded integers from the range of all possible options or one-hot encoding.

   This will lead to even a longer vector of parameters for the cost function (especially the more suited one-hot encoding version), but in section 6.2.1 we observed that PSO quickly 'learns' to ignore a set of parameters that are tied to some zeroed one.
3. When the optimization using the approach *2* takes too long, both approches can be combined in two phases:
   (a) apply the approach *1* (features with their default parameters only) to get a list of most important features,
   (b) only on those pre-selected features from the first phase, apply the approach *2* and fine-tune the feature parameters.

The main issue is that the number of input parameters to the optimized cost function can be too large leading to too slow optimization. To handle this, two-phase solution described in *3* could significantly reduce the dimensionality and thus the optimization time while still producing a good representation of the original data.

To evaluate the affect of the extracted features, we need some cost function consuming our vector of encoded features. This cost function will decode the vector, extract the features accordingly and utilize metrics to measure the importance of each extracted feature. Couple of metrics are possible here:

- *XGBoost*[2] - a software library offering an implementation of stochaistic gradient boosting algorithm. When fitted to the data, it provides feature importance scores from range $< 0; 1 >$. The individual scores can be simply summarized, retracted from the count of all possible extracted features and thus used as the output of the cost function which is a subject of minimalization.
- *Principal Component Analysis*: since our features and their parameters are encoded as numbers, we can measure their importance by looking at PCA components. Here, we can sum the largest absolute values of the Eigenvectors' components and proceed as in case of XGBoost. The reverse mapping from principal components to features is possible directly from the index of largest absolute values of the Eigenvectors' components.

This approach should be easy to implement and general-enough to be applicable to basically every data set and should produce the best-fitting extracted features for it.

---

[2] XGBoost library: `https://xgboost.readthedocs.io/`

Chapter 7

Case Study: Detection of *Nova Cas 2021*

In a world of automated sky surveys that monitors the sky continuously every night and producing huge amount of data, we were interested if our already trained models for light curves classification can be somehow useful. These automated sky surveys can provide light curve for basically every star in its field of view and our classifier (although trained on light curves from other instruments) should be able to perform its automated and fast classification to one of the star variability classes (if any). But it can do even more: under normal circumstances, star variability class should not change, unless something extraordinary happens. Such extraordinary phenomena could be for example the eruption of a star as a nova[1] which might result into a change in the predicted class in compare to its older flux. We decided to test this hypothesis.

Nova Cas 2021 is a nova that erupted in March 2021[2] and was formerly known as a *EW*-class variable star named in e.g. *Czech Variable star catalogue*[3] as *CzeV3217*. After the eruption as nova, we recognize it as *Nova Cas 2021* or *V1405 Cas* of class *EW+N*.

We were able to obtain the magnitude and flux for the star from its pre-nova era from the ASAS-SN catalogue[4] which covered a period from December 19, 2020 to February 12, 2021. For the nova era of the star, we obtained the magnitude by technique of web scrapping from AAVSO[5]

---

[1] In close binary star systems a smaller star (white dwarf type) pulls the hydrogen from its larger companion by its gravitational force. When reaching the critical mass, the termonuclear reaction ignites which involves rapid brightening of the star (several magnitudes in compare to the original brightness) that can last for several months.

[2] Alert Notice 735: Nova in Cassiopeia: N Cas 2021: `https://www.aavso.org/aavso-alert-notice-735`

[3] Czech Variable star catalogue: `http://var.astro.cz/`

[4] ASAS-SN data for *CzeV3217*: `https://asas-sn.osu.edu/sky-patrol/coordinate/00745124-4307-4bd9-9d95-d59d5399975e`

[5] AAVSO data for *V1405 Cas*: `https://app.aavso.org/webobs/results/?star=000-BNX-642&num_results=200&page=1`

which covers a period from March 17, 2021 to May 4, 2021.

Based on the formula from *GNU Astronomy Utilities manual*[6] we calculated the flux value F for the nova from its magnitude m, taking the reference values F_r and m_r from the pre-nova data (more specifically for the timestamp 2021-01-05.2318521 which has low errors in magnitude and flux). The resulting formula is:

$$F = \frac{10^{-\frac{(m-m_r)}{2.5}}}{F_r} \tag{7.1}$$

Giving $m_r = 15.305$ and $F_r = 2.741$ we then get the final formula for calculating the flux F knowing the magnitude m:

$$F = \frac{10^{-\frac{(m-15.305)}{2.5}}}{2.741} \tag{7.2}$$

The resulting light curves are depicted on Fig. 7.1 and 7.2. For the next steps we always applied data pre-processing steps for Kepler K2 multi-class classification (see section 6.2). First, we let our pre-trained Kepler K2 multi-class classifier to predict the class of the pre-nova light curve, which resulted into *RRab* class prediction. This is of course different from the correct one (*EW*), but as the classifier was trained on quite different data, it is not that important. Our hypothesis was that this class prediction should change when the nova erupted. To test this, we combined both the light curves into one, performed the same data pre-processing and let the pre-trained classifier to make a prediction, which resulted into *OTHPER* class that evokes some variability in the data with unknown phenomena (since our classifier was not trained for *EW+N* data, we consider this classification to be quite accurate). As the predicted class changed, our hypothesis was thus confirmed.

After this, we were interested how quickly our pre-trained classifier reacts on a change in the data and changes the predicted class. We discovered that 3 days after the eruption our classifier pre-trained for totally different light curves detects this change and changes the class prediction (see Fig. 7.3 for resulting light curve - you can also observe some typical signs of light curves: sparseness, different sampling rates and various instruments as data sources). On Fig. 7.4 you can see what portion of the available post-eruption data the classifier needed to change the predicted class (the red part represents the data of the first cca 3 days).

7.1 Possible Practical Use

These results can have interesting implications as it depicts that our classifier pre-trained for different light curves (coming from totally different instrument - Kepler is placed in the

---

[6] GNU Astronomy Utilities manual: `https://www.gnu.org/software/gnuastro/manual/html_node/Magnitude-to-flux-conversion.html`

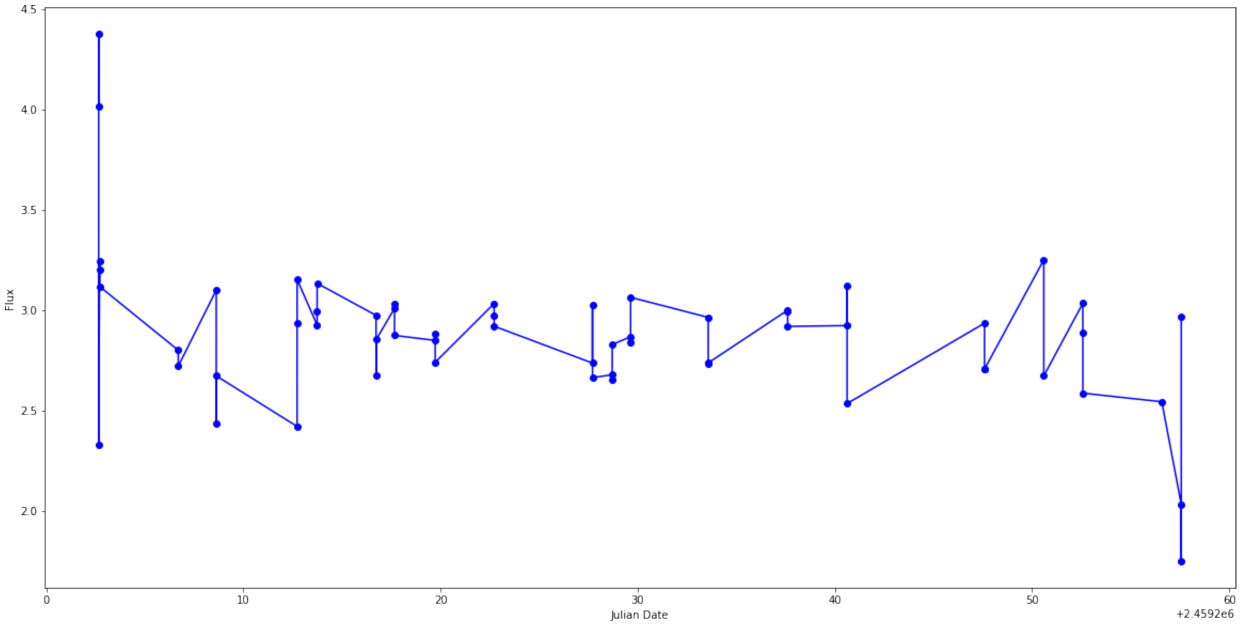


Figure 7.1: Light curve of variable star *CzeV3217* before its eruption as nova. We can observe its flux variability (y axis) during the time (Julian Date on x axis), but for all the time the changes are constant and around the mean value.

Universe while light curves of *CzeV3217* were obtained by Earth telescopes) can be used to at least detect interesting astronomical phenomena by changing the class prediction. This can be used for example for automated sky surveys producing light curves periodically (or at least often enough) as an automated and fast way of interesting phenomena detector within their pipelines. Our classifier also seems to be robust enough to handle light curves combined from multiple instruments to detect such change. To our knowledge there is no related work that attempted to perform such experiment.

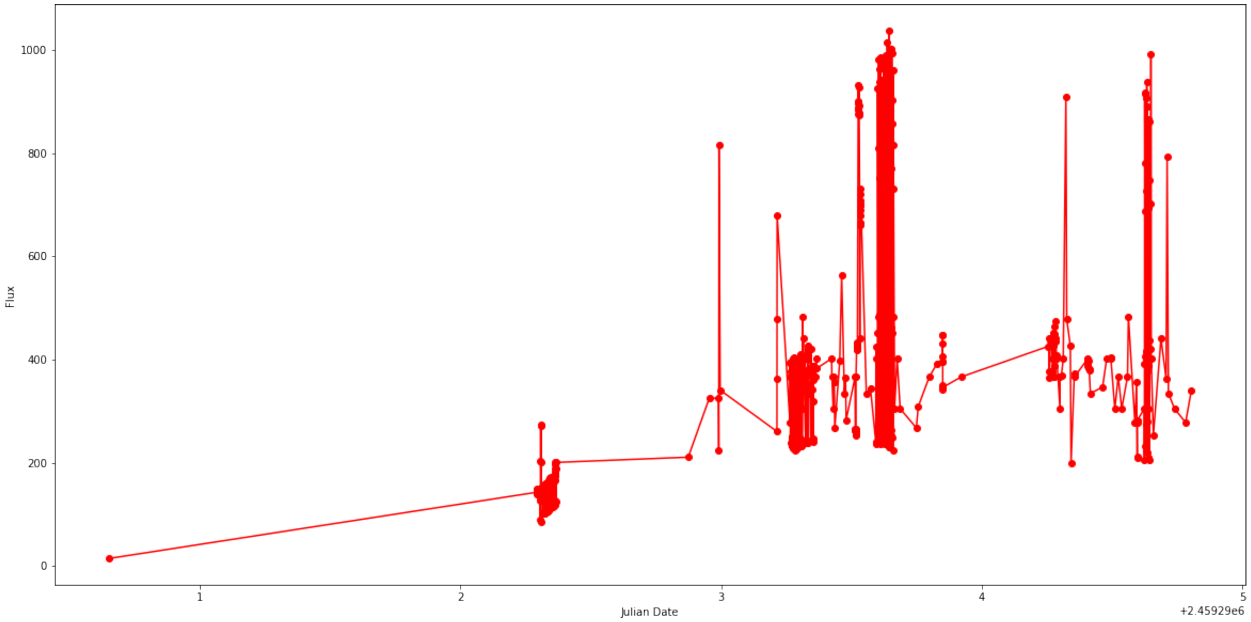


Figure 7.2: Light curve of variable star *CzeV3217* after its eruption as nova. We can observe different scale of the flux (y axis) in compare to its pre-nova values on Fig. 7.1.

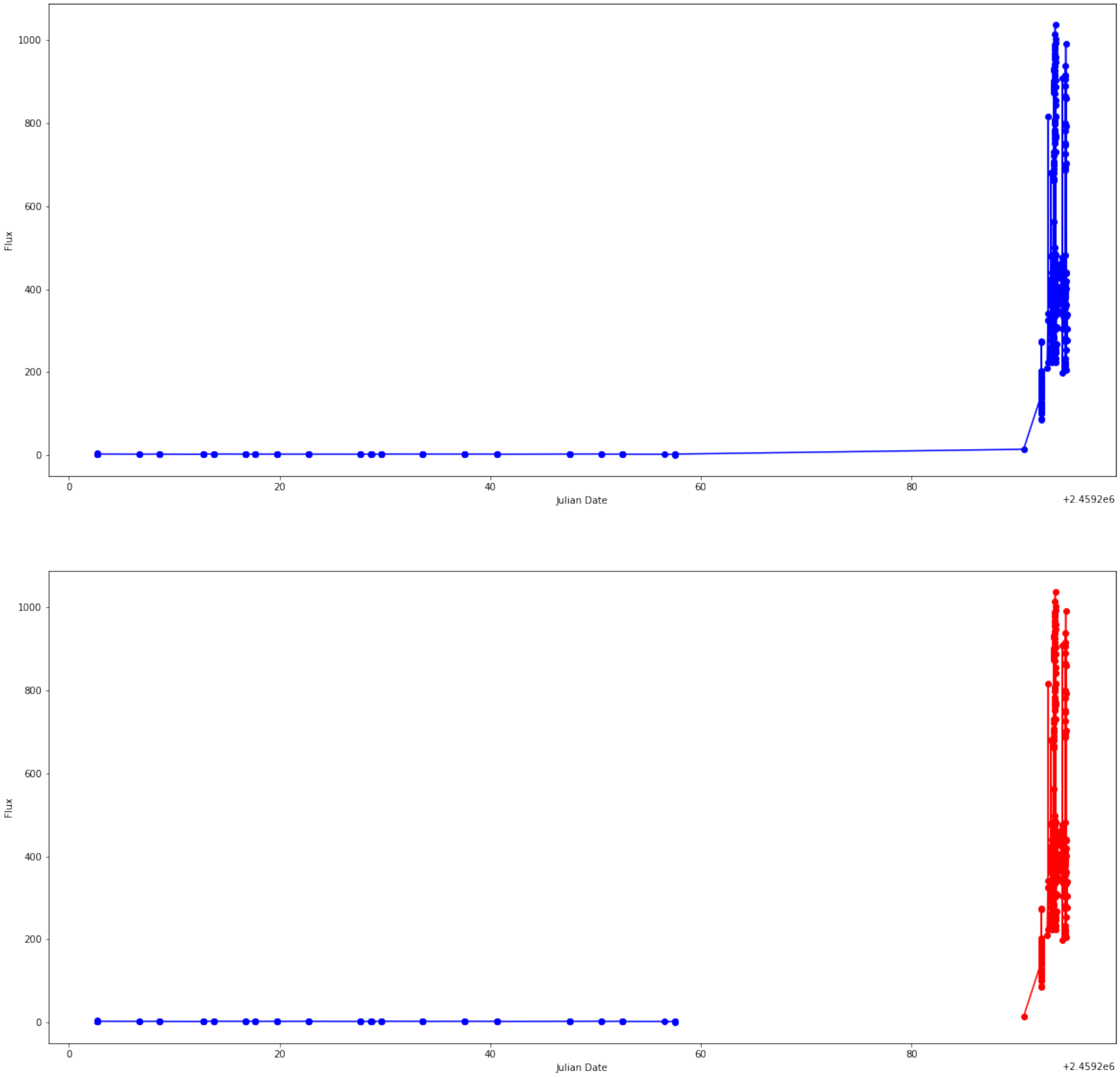


Figure 7.3: Combined light curves of variable star *CzeV3217* before and 3 days after its eruption as nova. On top we can see the light curve that our classifier used for prediction, on bottom there are both light curves plotted separatly.

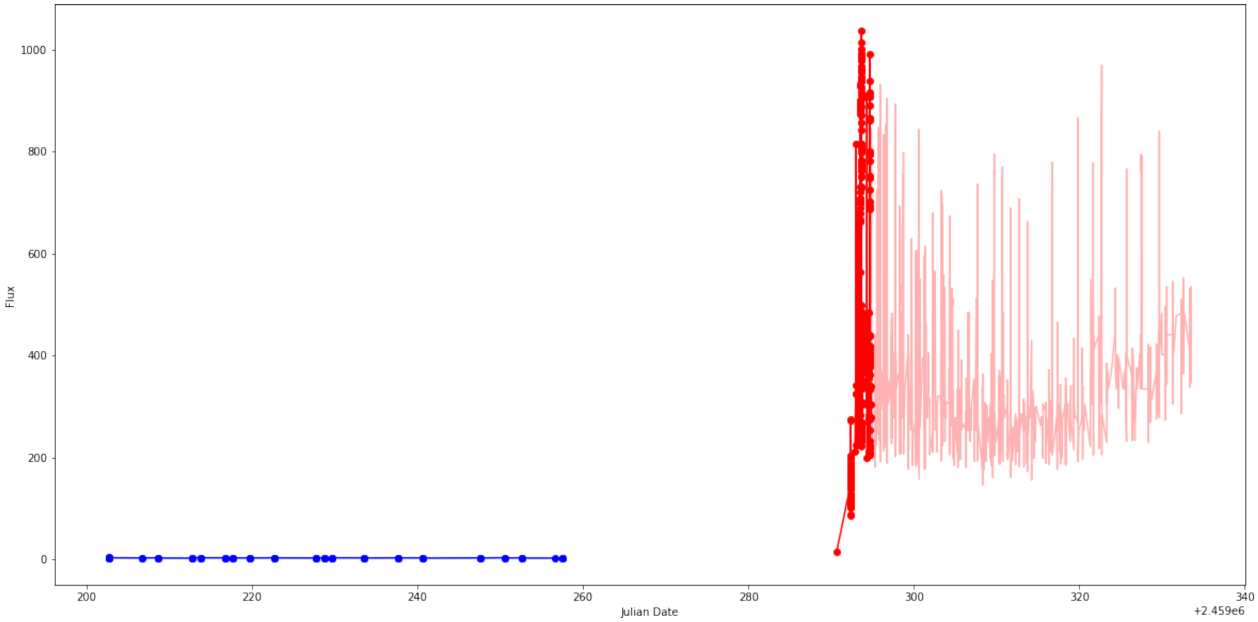


Figure 7.4: Combined light curves of variable star *CzeV3217* before and after its eruption as nova with the first 3 days after the eruption was spotted highlighted in red.

Chapter 8

Conclusions

Astronomical time series (a.k.a. light curves) are very specific in compare to other ones - they are noisy, variable in length and observational intervals and with overlapping classes. Traditional methods for their classification seems to not perform very well with them. Significant attributes extraction that includes various statistical markers and FFT coefficients seems to be necessary to achieve good results and the classification accuracy can be highly improved by optimizing the hyper-parameters with evolutionary algorithm. It also seems that successful classification model must contain a high number of hidden layers ("deep learning"), low training interval (in case of MLP that is also a good choice for classification based on extracted features) and data with high precision. Also a structure of hidden layers seems to have an impact on classification accuracy as we observed small number of neurons in layers at the beginning of the network and high number of neurons among the last layers of the network.

Our approach is time consuming, but good classifiers can be found very quickly during first iterations and then they are just fine-tuned. In the end it can reach high accuracy confirmed by validation and test set to prevent over-fitting. So far we are not aware of a related work reaching similar accuracy on Kepler K2 data set in both binary and multi-class classification. Similar work usually performs multi-class classification with help of photometric metadata as additional features while we rely just on the time series themselves (although in some experiments we select only those with high confidence in label, which seems to be especially important for binary classification - it seems that samples of individual classes may differ too much from each other to be united as one 'variable' class). No other work using Kepler or Kepler K2 data that we are aware of also mentions what kind of or how much data they used (that is, whether the data set was balanced, what was the number of samples used or what was the minimum or mean of the label confidence).

We expected that by introducing photometric meta-data we can lower the confidence in label while keeping similar accuracy. Instead it turned out that photometric metadata (at least without further pre-processing) brings just more dimensions to our approach but we can see no improvement in classification accuracy. Our classifier for Kepler K2 performs best without them - just with the light curves themselves.

We tested our approach on original Kepler data as well and while it seems that in some experiments it performs reasonably well (binary classification) then in other experiments we got rather poor results (multi-class classification). Further analysis suggests that this is most probably because of the set of extracted features originally established for Kepler K2 data set. Nevertheless we could observe a continuous improvement of the classification accuracy during evolutionary algorithm epochs so the design itself worked.

In all experiments we could observe that deep networks with batch normalization among layers performed best. The batch normalization with high momentum seems to be especially crucial for binary classification (both one-variability-class-specific and variable/non-variable). On the other hand, dropout was always zeroed.

Our last experiment with nova detection suggests that our approach is robust enough to work with light curves from other instruments (using a model pre-trained on a different data set). We demonstrated its usability as a fast, real-time anomaly detection (by watching for changes in predicted classes) in real-world light curves combined from various instruments and time epochs. At the time of writing, we are not aware of any related work in this area.

As a further work, we would like to experiment with our idea of evolutionary established extracted features as a mean of making our approach universal and maybe even automated. We would also like to address the convergence speed, where despite the improvement by modifying the cost function, it still remains challenging although we can see that good classifiers are found quite quickly. We are aware of papers that attempts to speed up the convergence of hyperparameters optimization using e.g. Decision trees or Bayesian optimization. A nice overview of optimizing (deep) neural networks also offers e.g. [50]. Other possibility for binary classification could be for example a solution inspired by anomaly detection where the anomaly could be anything but noise (where the noise simply masks a non-variable star and everything else would be considered variable).

Bibliography

1. SKODA, Petr. Optical Spectroscopy with the Technology of Virtual Observatory. *Baltic Astronomy*. 2011-12, vol. 20. Available from DOI: `10.1515/astro-2017-0332`.
2. ANDREŠIČ, D.; ŠALOUN, P.; SUCHÁNOVÁ, B. Large Astronomical Time Series Preprocessing and Visualization for Classification using Artificial Neural Networks. In: *2019 IEEE 15th International Scientific Conference on Informatics*. 2019, pp. 000311–000316. Available from DOI: `10.1109/Informatics47936.2019.9119283`.
3. HU, Bing; CHEN, Yanping; KEOGH, Eamonn J. Time Series Classification under More Realistic Assumptions. In: *SDM*. 2013.
4. PETITJEAN, F.; FORESTIER, G.; WEBB, G. I.; NICHOLSON, A. E.; CHEN, Y.; KEOGH, E. Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification. In: *2014 IEEE International Conference on Data Mining*. 2014-12, pp. 470–479. ISSN 2374-8486. Available from DOI: `10.1109/ICDM.2014.27`.
5. KARIM, Fazle; MAJUMDAR, Somshubra; DARABI, Houshang; CHEN, Shun. LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access*. 2018, vol. 6, pp. 1662–1669. ISSN 2169-3536. Available from DOI: `10.1109/access.2017.2779939`.
6. DAU, Hoang Anh; BAGNALL, Anthony; KAMGAR, Kaveh; YEH, Chin-Chia Michael; ZHU, Yan; GHARGHABI, Shaghayegh; RATANAMAHATANA, Chotirat Ann; KEOGH, Eamonn. *The UCR Time Series Archive*. 2018. Available from arXiv: `1810.07758 [cs.LG]`.
7. BAGNALL, Anthony; LINES, Jason; BOSTROM, Aaron; LARGE, James; KEOGH, Eamonn. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*. 2016-11, vol. 31, no. 3, pp. 606–660. Available from DOI: `10.1007/s10618-016-0483-9`.
8. FAWAZ, Hassan Ismail; FORESTIER, Germain; WEBER, Jonathan; IDOUMGHAR, Lhassane; MULLER, Pierre-Alain. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*. 2019-03, vol. 33, no. 4, pp. 917–963. Available from DOI: `10.1007/s10618-019-00619-1`.

9. SAKOE, H.; CHIBA, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. 1978-02, vol. 26, no. 1, pp. 43–49. ISSN 0096-3518. Available from DOI: `10.1109/TASSP.1978.1163055`.

10. SCHÄFER, Patrick. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*. 2014-09, vol. 29, no. 6, pp. 1505–1530. Available from DOI: `10.1007/s10618-014-0377-7`.

11. SART, Doruk; MUEEN, Abdullah; NAJJAR, Walid; KEOGH, Eamonn; NIENNATTRAKUL, Vit. Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs. In: *2010 IEEE International Conference on Data Mining*. 2010, pp. 1001–1006. Available from DOI: `10.1109/ICDM.2010.21`.

12. ELMAN, Jeffrey L. Finding Structure in Time. *Cognitive Science*. 1990-03, vol. 14, no. 2, pp. 179–211. Available from DOI: `10.1207/s15516709cog1402_1`.

13. BAGNALL, A.; LINES, J.; HILLS, J.; BOSTROM, A. Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. *IEEE Transactions on Knowledge and Data Engineering*. 2015-09, vol. 27, no. 9, pp. 2522–2535. ISSN 2326-3865. Available from DOI: `10.1109/TKDE.2015.2416723`.

14. ŠKODA, P.; PODSZTAVEK, O.; TVRDIÉK, P. Active deep learning method for the discovery of objects of interest in large spectroscopic surveys. *Astronomy & Astrophysics*. 2020-11, vol. 643, pp. A122. Available from DOI: `10.1051/0004-6361/201936090`.

15. LECUN, Yann; BENGIO, Y. Convolutional Networks for Images, Speech, and Time-Series. In: 1995-01.

16. CUI, Zhicheng; CHEN, Wenlin; CHEN, Yixin. *Multi-Scale Convolutional Neural Networks for Time Series Classification*. 2016. Available from arXiv: `1603.06995 [cs.CV]`.

17. RUMELHART, David E. chapter Parallel Distributed Processing, Exploration in the Microstructure of Cognition. In: 1986.

18. SAHA, Rohan. Comparing Classification Models on Kepler Data. 2019. Available from DOI: `10.13140/RG.2.2.11232.43523`.

19. LECUN, Yann; BOSER, Bernhard E.; DENKER, John S.; HENDERSON, Donnie; HOWARD, R. E.; HUBBARD, Wayne E.; JACKEL, Lawrence D. Handwritten Digit Recognition with a Back-Propagation Network. In: TOURETZKY, D. S. (ed.). *Advances in Neural Information Processing Systems 2*. Morgan-Kaufmann, 1990, pp. 396–404. Available from DOI: `10.5555/109230.109279`.

20. LECUN, Yann; HAFFNER, Patrick; BOTTOU, Léon; BENGIO, Yoshua. Object Recognition with Gradient-Based Learning. In: *Shape, Contour and Grouping in Computer Vision*. Springer Berlin Heidelberg, 1999, pp. 319–345. Available from DOI: `10.1007/3-540-46805-6_19`.

21. FUKUSHIMA, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*. 1980-04, vol. 36, no. 4, pp. 193–202. Available from DOI: `10.1007/bf00344251`.

22. SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*. 2014-09, vol. 4. Available from DOI: `10.5555/2969033.2969173`.

23. BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. Available from arXiv: `1409.0473 [cs.CL]`.

24. GAMBOA, John Cristian Borges. *Deep Learning for Time-Series Analysis*. 2017. Available from arXiv: `1701.01887 [cs.LG]`.

25. WANG, Zhiguang; YAN, Weizhong; OATES, Tim. *Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline*. 2016. Available from DOI: `10.1109/IJCNN.2017.7966039`.

26. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition*. 2015. Available from DOI: `10.1109/CVPR.2016.90`.

27. SMIRNOV, Denis; NGUIFO, Engelbert Mephu. Time Series Classification with Recurrent Neural Networks. In: 2018.

28. HOCHREITER, S. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München*. 1991.

29. RUSSAKOVSKY, Olga et al. *ImageNet Large Scale Visual Recognition Challenge*. 2014. Available from DOI: `10.1007/s11263-015-0816-y`.

30. WILLIAMS, R. J.; ZIPSER, D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*. 1989-06, vol. 1, no. 2, pp. 270–280. ISSN 0899-7667. Available from DOI: `10.1162/neco.1989.1.2.270`.

31. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. *Neural Computation*. 1997-11, vol. 9, no. 8, pp. 1735–1780. Available from DOI: `10.1162/neco.1997.9.8.1735`.

32. HINNERS, Trisha A.; TAT, Kevin; THORP, Rachel. Machine Learning Techniques for Stellar Light Curve Classification. *The Astronomical Journal*. 2018-06, vol. 156, no. 1, pp. 7. ISSN 1538-3881. Available from DOI: `10.3847/1538-3881/aac16d`.

33. GILES, C. Lee; LAWRENCE, Steve; TSOI, Ah Chung. *Machine Learning*. 2001, vol. 44, no. 1/2, pp. 161–183. Available from DOI: `10.1023/a:1010884214864`.

34. E., VAN CLEVE J.; A., CALDWELL D. *Kepler: A Search for Terrestrial Planets, KSCI-19033-002* [Kepler Science Document]. 2016.

35. BASS, Gideon; BORNE, Kirk. Supervised Ensemble Classification of Kepler Variable Stars. *Monthly Notices of the Royal Astronomical Society*. 2016-04, vol. 459, pp. stw810. Available from DOI: `10.1093/mnras/stw810`.

36. HOSENIE, Zafiirah; LYON, Robert J; STAPPERS, Benjamin W; MOOTOOVALOO, Arrykrishna. Comparing Multiclass, Binary, and Hierarchical Machine Learning Classification schemes for variable stars. *Monthly Notices of the Royal Astronomical Society*. 2019-07, vol. 488, no. 4, pp. 4858–4872. ISSN 1365-2966. Available from DOI: `10.1093/mnras/stz1999`.

37. SAMUS', N. N.; KAZAROVETS, E. V.; DURLEVICH, O. V.; KIREEVA, N. N.; PASTUKHOVA, E. N. General catalogue of variable stars: Version GCVS 5.1. *Astronomy Reports*. 2017-01, vol. 61, no. 1, pp. 80–88. Available from DOI: `10.1134/s1063772917010085`.

38. ARMSTRONG, D. J.; OSBORN, H. P.; BROWN, D. J. A.; KIRK, J.; LAM, K. W. F.; POLLACCO, D. L.; SPAKE, J.; WALKER, S. R. *K2 Variable Catalogue I: A Catalogue of Variable Stars from K2 Field 0*. 2014. Available from arXiv: `1411.6830 [astro-ph.SR]`.

39. ARMSTRONG, D. J.; KIRK, J.; LAM, K. W. F.; MCCORMAC, J.; WALKER, S. R.; BROWN, D. J. A.; OSBORN, H. P.; POLLACCO, D. L.; SPAKE, J. K2 Variable Catalogue: Variable stars and eclipsing binaries in K2 campaigns 1 and 0. *Astronomy & Astrophysics*. 2015-06, vol. 579, pp. A19. ISSN 1432-0746. Available from DOI: `10.1051/0004-6361/201525889`.

40. CLEVE, Jeffrey Edward van et al. Kepler: A Search for Terrestrial Planets - Kepler Data Characterization Handbook. In: 2016.

41. JENKINS, Jon M. *Kepler Data Processing Handbook: Overview of the Science Operations Center* [Kepler Science Document]. 2017-01.

42. VANDERBURG, Andrew. *K2 Extracted Lightcurves ("K2SFF")*. STScI/MAST, 2015. Available from DOI: `10.17909/T9BC75`.

43. SAMMON, J.W. A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*. 1969-05, vol. C-18, no. 5, pp. 401–409. Available from DOI: `10.1109/t-c.1969.222678`.

44. BATISTA, Gustavo E. A. P. A.; KEOGH, Eamonn J.; TATAW, Oben Moses; SOUZA, Vinícius M. A. de. CID: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*. 2013-04, vol. 28, no. 3, pp. 634–669. Available from DOI: `10.1007/s10618-013-0312-3`.
45. SCHREIBER, Thomas; SCHMITZ, Andreas. Discrimination power of measures for nonlinearity in a time series. *Physical Review E*. 1997-05, vol. 55, no. 5, pp. 5443–5447. Available from DOI: `10.1103/physreve.55.5443`.
46. BOX, George E. P.; JENKINS, Gwilym M.; REINSEL, Gregory C. *Time Series Analysis*. Wiley, 2008-06. Available from DOI: `10.1002/9781118619193`.
47. MAATEN, Laurens van der; HINTON, Geoffrey. Visualizing Data using t-SNE. *Journal of Machine Learning Research*. 2008, vol. 9, pp. 2579–2605. Available also from: `http://www.jmlr.org/papers/v9/vandermaaten08a.html`.
48. IOFFE, Sergey; SZEGEDY, Christian. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. Available from DOI: `10.5555/3045118.3045167`.
49. KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995-11, vol. 4, 1942–1948 vol.4. ISSN null. Available from DOI: `10.1109/ICNN.1995.488968`.
50. TALBI, El-Ghazali. *Optimization of deep neural networks: a survey and unified taxonomy*. 2020-06. Available also from: `https://hal.inria.fr/hal-02570804`. working paper or preprint.

Appendix A

List of Publications

A.1 Author's Bibliography Related to the Thesis

[1] D. Andrešič, P. Šaloun and I. Anagnostopoulos, "Efficient big data analysis on a single machine using apache spark and self-organizing map libraries," 2017 12th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), Bratislava, 2017, pp. 1-5. (SJR: 0.194)

[2] D. Andrešič, P. Šaloun, "Efektivní analýza velkých dat pomocí Apache Spark a samoučících neuronových sítí na jediném počítači," Data a znalosti 2017, Plzeň, 2017.

[3] D. Andrešič, P. Šaloun, "Effective Big Data Analysis on single-node Apache Spark with Self-Organizing Map Libraries," WOFEX 2017, Ostrava, 2017.

[4] D. Andrešič, P. Šaloun, B. Pečíková, "Strojové učení při analýze rozsáhlých astronomických datasetů s časovými řadami," Data a znalosti & WIKT 2018, Brno, 2018.

[5] D. Andrešič, P. Šaloun, B. Pečíková, "Large Astronomical Time Series Pre-processing and Visualization for Classification using Artificial Neural Networks," 2019 IEEE 15th International Scientific Conference on Informatics, Poprad, 2019. (SJR: 0.157)

[6] D. Andrešič, P. Šaloun, B. Pečíková, "Large Astronomical Time Series Pre-processing for Classification Using Artificial Neural Networks," 2021 In: J. Paralič, P. Sinčák, P. Hartono, & V. Mařík (Eds.), Advances in Intelligent Systems and Computing. Springer International Publishing. https://doi.org/10.1007/978-3-030-63872-6 (SJR: 0.184; Q3)

[7] D. Andrešič, P. Šaloun, B. Pečíková, "Large Astronomical Time Series Pre-processing for Classification Using Artificial Neural Networks," 2021 In: Zelinka I., Brescia M., Baron D. (eds) Intelligent Astrophysics. Emergence, Complexity and Computation, vol 39. Springer, Cham. https://doi.org/10.1007/978-3-030-65867-0_12

A.2 Author's Bibliography Unrelated to the Thesis

[1] Šaloun, Petr & Andrešič, David & Skoda, Petr & Zelinka, Ivan. (2016). Time Series, Collaboration and Large Data Sets Enhancements of SPLAT-VO. 111-116. 10.1109SIMS.2016.20. (SJR: 0.129)

[2] D. Andrešič, A. Ondrejka, P. Šaloun and R. Ceplákova, "Webový portál pro identifikaci poruchy osobnosti z psaného textu," 12th Workshop on Intelligent and Knowledge oriented Technologies 2017, Košice, 2017.

[3] P. Šaloun, M. Malčik, D. Andrešič and D. Nespěšný, "Using eyetracking to analyse how flowcharts are understood," 2017 IEEE 14th International Scientific Conference on Informatics, Poprad, 2017, pp. 394-399. (SJR: 0.157)

[4] D. Andrešič, P. Šaloun, B. Cigánková, "Crowd Sourcing as an Improvement of N-Grams Text Document Classification Algorithm," WOFEX 2018, Ostrava, 2018.

[5] D. Andrešič, P. Šaloun, B. Cigánková, "Vylepšení klasifikace textových dokumentů algoritmem N-Grams pomocí crowdsourcingu," Data a znalosti & WIKT 2019, Košice, 2019.

[6] P. Šaloun, D. Andrešič, B. Cigánková and I. Anagnostopoulos, "Crowd Sourcing as an Improvement of N-Grams Text Document Classification Algorithm," 2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization (SMA, 2020, pp. 1-6, doi: 10.1109/SMAP49528.2020.9248454. (SJR: 0.194)

[7] Andrešič D., Šaloun P., Šaloun P., Mališů P., Dragon T., Vagner L. Umělá inteligence při výuce programování v době Průmyslu 4.0. In Malčík M. (Eds.) Vzdělávání ve Společnosti 4.0. 2020.

Appendix B

Author's Curriculum Vitae

B.1 Education

VŠB - Technical University of Ostrava

Ostrava, Czech Republic

Doctoral degree, Faculty of Electrical Engineering and Computer Science
09/2016 - ***now***

- Specialization: Computer Science, Communication Technology and Applied Mathematics
- Thesis topic: *Big Data Processing by Means of Unconventional Algorithm*
- Case study: Astronomical time series classification by means of deep artificial neural network (multi-layer perceptron) with topology established by evolutionary algorithm

Master's degree, Faculty of Electrical Engineering and Computer Science
09/2013 - 06/2016

- Specialization: Information and Communication Technology
- Thesis topic: *Programme for the Post-processing and Analysis of Complex Large-Scale Spectroscopic Surveys Using the Virtual Observatory Protocols*

Bachelor's degree, Faculty of Electrical Engineering and Computer Science
09/2010 - 06/2013

- Specialization: Information and Communication Technology

  - Thesis topic: *Programme for Interactive Spectra Analysis in Virtual Observatory Environment*

Střední průmyslová škola elektrotechniky a informatiky, Ostrava, příspěvková organizace

Ostrava - Moravská Ostrava, Czech Republic

Graduation Exam, Low-current Electrical
09/2003 - 06/2007

  - Specialization: Microprocessor Technology, Radioelectronics

B.2 Work Experience

Palacký University Olomouc

Olomouc, Czech Republic

2021 - now

  - Cooperation on education for foreign students ("Python for everyday use")

VŠB - Technical University of Ostrava

Ostrava, Czech Republic

2019 - now

  - Cooperation on TAČR project (CMP rádce) - creating a text document classifier that utilizes n-grams and TF/IDF approach for statistical classification

CertiCon a. s.

Prague, Czech Republic

Senior Software Engineer
2016 - now

  - Development of diagnostic software for automotive area
  - Development of ground/ground communication system for airports
  - Work in SCRUM as a part of international teams with English on a daily basis

- Experience as a technical leader
- Java desktop and network development
- Experience with a properietary cloud platform
- Active in internal Java training group
- Java, Eclipse Equinox, Akka, Docker, Apache Maven, OSGi, Jenkins

Etnetera a. s.

Prague, Czech Republic

Java Developer
2013 - 2016

- Development of an Internal System for a Large Automotive Corporation
- Development Time Estimatation
- Implementation Analysis and Design of Individual System Modules
- Responsibility for Project Documentation
- Responsibility for Project Maintain, Assembly and Deployment
- Java, PL/SQL, Hibernate, Apache Wicket, Spring, UML, Maven, Apache Tomcat, IBM WebSphere etc.

Astronomical Institute of the Czech Academy of Sciences

Ondřejov, Czech Republic

Java Programmer and Development Coordinator
2016

- Employment Agreement
- Enhancement of Spectral Analysis Tool and its Development Process
- Continuous Integration (Jenkins CI, GitHub Integration)
- Suggestions and Realization of Project Development Process Enhancements
- Active Communication in English within International Team
- Linux Server Administration (+ Docker, Jenkins CI, Apache2 as Reverse Proxy)
- Java, Ant, Jenkins CI, MediaWiki, Docker, Linux, Apache, GitHub

Astronomical Institute of the Czech Academy of Sciences

Ondřejov, Czech Republic

Java Programmer
2015

- Employment Agreement
- Enhancement of Spectral Analysis Tool Behind the Scope of Diploma Thesis
- Continuous Integration - Preparing Docker Image with Jenkins CI
- Suggestions of Project Development Process Enhancements (Usage of Github Issues, MediWiki etc.)
- Active Communication in English within International Team
- Java, Ant

NetDirect s.r.o.

Ostrava, Czech Republic

Programmer
2008 - 2013

- Development of E-commerce Solutions
- Development Time Estimatation
- Complete Project Analysis
- C#, .NET, XSLT, T-SQL, AJAX, jQuery, etc.

B.3 Certificates

Základní studijní program Machine Learning College

Oracle Certified Associate, Java SE 8 Programmer Oracle

Introduction to Big Data Coursera Course Certificates

Big Data Modeling and Management Systems Coursera Course Certificates

Big Data Integration and Processing Coursera Course Certificates

Machine Learning With Big Data Coursera Course Certificates

B.4 Computer skills

Programming languages Java (actively), Python (actively), C# (formerly), SQL (actively couple of years ago), C/C++, PHP, VBScript, Haskell, assembler

Databases MS SQL, Oracle, H2, MySQL, PostgreSQL

Web JSON, XHTML, JavaScript, AJAX, CSS, HTML5,

Frameworks Apache Wicket, Hibernate, OSGi, Spring, .NET, jQuery

Operating systems GNU/Linux (all mainstream distros, Linux From Scratch), Microsoft Windows

Development tools Eclipse, Intellij Idea, MS Visual Studio, NetBeans, Maven, Ant, Jenkins CI

Networking and system administration Basic Linux Gateway and Server (Iptables, NAT, Apache, Tomcat, PHP, MySQL, PostgreSQL, Samba), Docker, IBM WebSphere

Analytical and documentation tools UML, Basic Design Patterns, LaTeX, JavaDoc

B.5 Language Skills

English advanced

Czech native speaker