FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Cloud-Based Platform for Active Learning of Astronomical Spectra
Student:	MUDr. Tomáš Mazel Ph.D.
Supervisor:	RNDr. Petr Škoda, CSc.
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2019/20

Instructions

The goal of the thesis is the design of an engine based on current VO-CLOUD infrastructure, allowing to conduct different scenarios of active learning of millions of astronomical spectra in order to identify candidates with given spectral features and characteristic shapes. One of the crucial part is the interactive (re)classification/rejection of suggested candidates with the possibility to obtain all important meta-data and visualize them in different steps of processing, even in dimensionality reduced plots (e.g. PCA, tSNE).
1) Analyze the typical workflows of active learning.
2) Suggest ideal capabilities of such system and confront them with those available in VO-CLOUD.
3) Design the missing modules and control logic.
4) Implement the platform, taking into account also its easy deployability (consider Docker).
5) Discuss the user experience and performance of your solution and suggest possible future improvements.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 6, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Cloud-Based Platform for Active Learning of Astronomical Spectra

MUDr. Tomáš Mazel, Ph.D.

Department of Software Engineering
Supervisor: RNDr. Petr Škoda, CSc.

February 12, 2020

Acknowledgements

I would like to thank my supervisor Dr. Petr Škoda for his help and advice, my colleagues Jakub Koza and Ondřej Podstavek for their assistance, and my family for their long-lasting support, patience and encouragement.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that make use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on February 12, 2020 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Mazel, Tomáš. *Cloud-Based Platform for Active Learning of Astronomical Spectra*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Cílem této práce je poskytnout uživatelům VO-CLOUD systému možnost použít aktivní učení pro klasifikaci astronomických objektů na základě jejich spektrálních vlastností.

V rámci systému VO-CLOUD bylo vytvořeno webové rozhraní, kde uživatelé mohou aplikovat aktivní učení s využitím konvoluční neuronové sítě se zvolenými parametry. Program umožňuje vytvořit iniciální trénovací soubor a následně v jednotlivých iteracích uživateli předkládá sadu spekter, u nichž si byl klasifikační algoritmus neuronové sítě nejméně jistý, k manuální klasifikaci. Spolu s grafy původních i předzpracovaných spekter jsou uživateli zobrazována i jím zvolená metadata zahrnující údaje o daném astronomickém objektu a technické údaje o daném spektru. Výsledky klasifikace je možno spolu s poznámkami uložit do textového souboru a Elasticsearch databáze. Tato nově vytvořená databáze obsahuje základní údaje o řádově milionech spekter z projektu LAMOST (Large Sky Area Multi-Object Fibre Spectroscopic Telescope) i ondřejovské kolekce. V každé iteraci je rovněž vyhodnocena úspěšnost klasifikačního algoritmu. V případě, že přesáhne zvolenou hodnotu, tj. v případě, že parametry neuronové sítě jsou již dostatečně optimalizovány, je možno aktivní učení ukončit a zobrazit nalezené zajímavé objekty z vybraných klasifikačních tříd.

Uživatelé VO-CLOUD systému budou mít nyní možnost interaktivně klasifikovat astronomické objekty pomocí algoritmů aktivního učení.

Klíčová slova astronomická spektra, webová platforma, interaktivní klasifikace, strojové učení, hluboké učení, konvoluční neuronové sítě, virtuální observatoř, VO-CLOUD, astroinformatika, Elasticsearch

Abstract

The aim of this work is to provide VO-CLOUD system users the option to use active learning methods to classify astronomical objects based on their spectral properties.

A web-based interface integrated into the VO-CLOUD system has been designed allowing users to apply active learning algorithms with user-defined parameters using an underlying convolutional neural network. The software allows to create the initial training set and then in iterations presents a set of spectra where the classification algorithm had the highest uncertainty for manual classification by the user. Along with the graphs of the original and preprocessed spectra, the user is presented with preselected set of metadata including information about the astronomical object and technical information about the displayed spectra. Classification results together with comments may be saved in a text file and an Elasticsearch database. This newly created database contains basic information about millions of spectra from the LAMOST (Large Sky Area Multi-Object Fibre Spectroscopic Telescope) project, as well as the Ondřejov collection. In each iteration the performance of the classification algorithm is evaluated and, in case it reaches a predefined value, i.e. in case the parameters of the neural network are sufficiently optimized, the active learning process may be stopped and objects belonging to classification classes of interest may be displayed.

In summary, VO-CLOUD users will now have the possibility to interactively (re)classify astronomical objects using active learning algorithms.

Keywords astronomical spectra, web platform, interactive classification, machine learning, deep learning, convolutional neural networks, virtual observatory, VO-CLOUD, astroinformatics, Elasticsearch

Contents

List of Figures

List of Tables

Introduction

With the advent of new methods and improved equipment in the field of astronomical spectrometry there is increasing need to analyze large amounts of spectroscopy data. Automated analysis using artificial intelligence methods will play an ever-increasing role.

This work intends to provide an interface allowing users of the virtual observatory VO-CLOUD software system to apply active learning methods to interactively classify astronomical spectra. Particular attention is given to convolutional neural networks (CNN), one of the most commonly used deep learning methods.

The thesis extends previous work of several bachelor and master students, who contributed to the design of the VO-CLOUD platform [1, 2, 3, 4, 5, 6].

In the first part of my thesis, I present an analysis of the problem and discuss available approaches. Then I present the solution and discuss its properties, performance and possible future developments.

Chapter 1

Aims and Objectives

The goal of the thesis is to design an engine based on current VO-CLOUD infrastructure allowing to conduct different scenarios of active learning of astronomical spectra in order to identify candidates with given spectral features and characteristic shapes. One of the crucial parts is the interactive (re)classification/rejection of suggested candidates with the possibility to obtain all important metadata and visualize them in different steps of processing.

The first objective is to analyze the typical workflow of active learning, suggesting ideal capabilities of such a system and confronting them with those available in VO-CLOUD.

Practical goals include designing missing modules and control logic and implementing the platform, testing performance of the solution and suggesting possible future improvements.

1.1 Functional Requirements

FR1: The active learning platform should be incorporated into the VO-CLOUD system as a new type of task/job.

FR2: Users should be able to define the active learning task features in form of a JSON configuration script.

FR3: Most commonly used JSON configurations should be predefined in form of JSON configuration files stored in the VO-CLOUD system.

FR4: Users should be able to upload the data to be analyzed to the VO-CLOUD server.

FR5: Users should have the option to prepare the initial training set by labelling randomly selected samples from the data pool to be analyzed.

FR6: Users should have the option to prepare the initial training set by labelling samples from a dataset different from the data pool to be analyzed.

FR7: Users should have the option to prepare the initial training set by labelling samples from user-selected original data.

FR8: Users should be able to visualize the spectral profile of celestial objects of interest with optional zoom and display of values on mouseover. They should have the option to visualize both the original record and the normalized spectrum in the wavelength region of interest.

FR9: Users should be able to see metadata corresponding to the visualized spectra.

FR10: Users should be able to manually label individual samples. Keyboard shortcuts should be considered in order to make labelling user-friendly and efficient.

FR11: In each iteration, samples for teacher's labelling should be selected based on the degree of uncertainty of their classification — the most uncertain samples first.

FR12: In each iteration, randomly selected samples should also be presented for evaluation by the teacher/expert/oracle in order to estimate performance accuracy of the algorithm. A predefined level of performance accuracy is typically the necessary condition for stopping the active learning procedure.

FR13: Users should be able to see performance accuracy of their labelling.

FR14: Users should be able to save and download the results of the labelling session.

FR15: Users should be able to save and download the predictions of each active learning iteration.

FR16: Users should be able to save and download the uncertainties associated with the predictions of each active learning iteration.

FR17: Users should be able to visualize the predicted spectra of interest.

1.2 Non-Functional Requirements

NFR1: A database containing metadata from LAMOST (Large Sky Area Multi-Object Fiber Spectroscopic Telescope) archive should be established.

NFR2: The database should allow very fast full-text search.

NFR3: The database should allow addition of new data categories such as labelling results.

NFR4: The database design should minimize requirements on resources (especially storage space).

Chapter 2

Analysis of the Problem

In this section I provide a short introduction to astronomical spectroscopy, stellar classification and Be stars, the main objects of interest in this study. Then I give a brief overview of active learning and convolutional neural networks. Finally, I present the VO-CLOUD project and its current state.

2.1 Astronomical Spectra

One of the main research areas of interest in the Astronomical Institute in Ondřejov is spectroscopical analysis of celestial objects. Electromagnetic spectra, typically represented in form of graphs of intensity of electromagnetic radiation as a function of wavelength or frequency, provide substantial part of what we know about the Universe, including information about surface temperature, chemical composition, distance and relative velocity of stellar, as well as non-stellar objects.

Foundations of optical spectroscopy were laid down at the beginning of the 18^{th} century by Isaac Newton when he used a prism to split the sunlight [7]. About one hundred years later Fraunhofer improved the optical devices significantly and for the first time observed discrete emission lines in fire light. Curious whether he could observe similar lines in the sunlight, he discovered the first dark absorption lines in the light of the Sun [8]. He then combined his prism with a telescope to observe the spectra of nearby planets and various stars such as Betelgeuse [9].

2.1.1 Continuous Spectra

In 1850s Kirchhoff and Bunsen offered an explanation of Fraunhofer's observations: hot solid objects produce light with a continuous spectrum, hot gases emitting light at specific wavelength are responsible for emission lines, while cool gases surrounding hot objects are responsible for the absorption lines [9, 10].

Thermal radiation responsible for the continuous part of the spectra is actually generated by all matter with a temperature greater than absolute zero. Gustav Kirchhoff introduced the term black body, an idealized opaque non-reflective object absorbing all incident electromagnetic radiation regardless of its frequency [11]. A black body in thermal equilibrium emits electromagnetic radiation, called black body radiation. The shape of its characteristic continuous radiation spectrum depends solely on temperature and was later explained by Max Planck in his famous Planck's law [12]:

$$B_\lambda = \frac{2hc^2}{\lambda^5(e^{\frac{hc}{\lambda k_B T}} - 1)}, \tag{2.1}$$

where B_λ represents spectral radiance (energy emitted by a given surface per unit solid angle, unit projected area, unit time, and unit frequency) measured in W.sr$^{-1}m^{-2}$,

h is Planck's constant (6.626 x 10^{-34}m^{2}kg/s,

c is the speed of light in vacuum (299 792 458 m/s),

k_B is Boltzmann constant (1.38 x 10^{-23}m^{2}kg s^{-2}K^{-1}), and

T is the temperature (in K).

Figure 2.1 shows black body radiation spectra at temperatures corresponding to major spectral classes of stars.

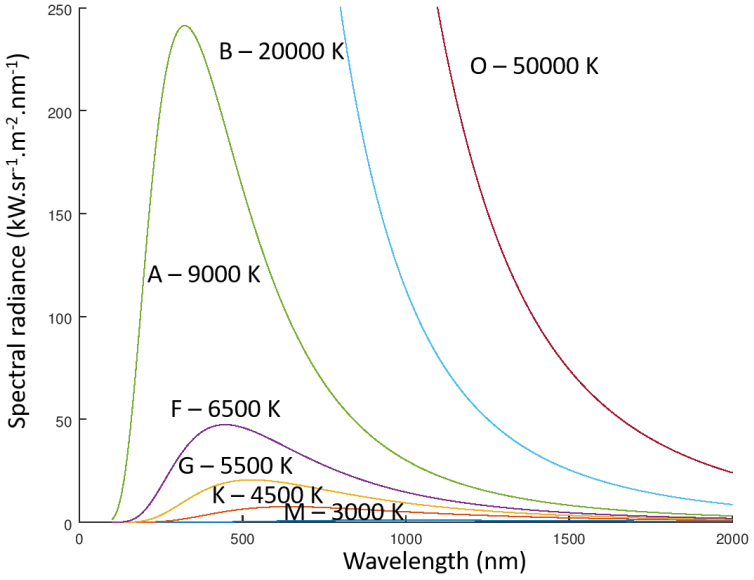

Figure 2.1: Black body radiation at temperatures typical for individual spectral classes of stars

Any real physical body emits a continuous spectrum similar in shape to that of the black body of the same temperature. The ratio between the real body and black body radiation is described by emissivity. Emissivity of real bodies ranges from 0 to 1 (corresponding to the ideal black body) [13].

Discrete spectral lines are typically related to transitions of electrons between individual energy levels [14]. Atoms, ions and molecules have their characteristic spectral lines [10, 15].

2.1.2 Spectral Classes

The most commonly used system of star classification is the Morgan-Keenan system [16] where stars are assigned a spectral class corresponding to the Harvard spectral classification system and a luminosity class [17]. The Harvard system groups the stars based on their surface temperature into classes O–M (see Table 2.1) [18].

Table 2.1: Harvard star classification system [17]

Class	Surface temperature	Color	Main sequence mass
O	over 30 000 K	blue	over 16 M_{Sun}
B	10 000 – 30 000 K	blue white	2.1 – 16 M_{Sun}
A	7 500 – 10 000 K	white	1.4 – 2.1 M_{Sun}
F	6 000 –7 500 K	yellow white	1.04 – 1.4 M_{Sun}
G	5 200 – 6 000 K	yellow	0.8 – 1.04 M_{Sun}
K	3 700 – 5 200 K	orange	0.45 – 0.8 M_{Sun}
M	2 400 – 3 700 K	orange–red	0.08 – 0.45 M_{Sun}

Luminosity classes are labelled I to VII, where class I corresponds to supergiants, II and III to giants, IV to subgiants and V to main sequence stars. Classes VI and VII represent subdwarfs and white dwarfs, respectively [17]. Luminosity is the total amount of electromagnetic energy emitted by the star (measured in watts) and is thus an absolute quantity in contrast to apparent brightness and star magnitude that depend on the distance between the observer and the star [19, 20].

2.1.3 Be Stars

Be stars are spectral B-type stars with emission lines [21]. Classical Be stars are non-supergiant, main sequence stars that have or once had one or more emission lines in Balmer series of hydrogen atom spectral lines resulting from electron transitions from levels with principal quantum number $n \geq 3$ to levels with $n=2$ [21]. As these lines are the most prominent features in hydrogen atom visible spectrum, they are referred to as Hα (656 nm), Hβ (486 nm)...

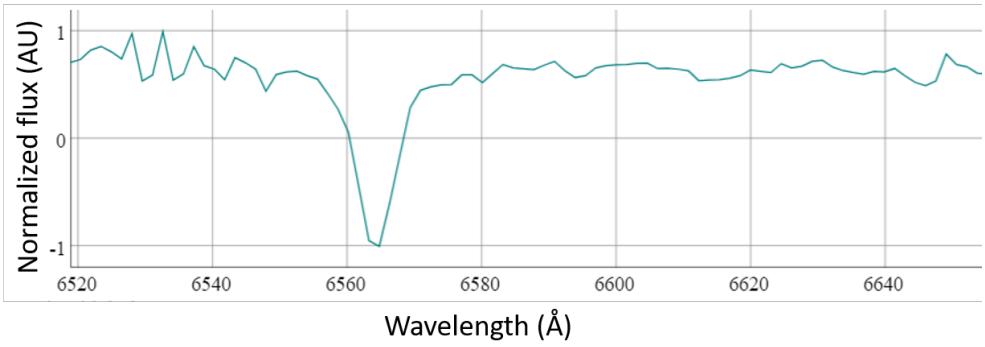

Figure 2.2: Example of an absorption peak from Ondřejov archive

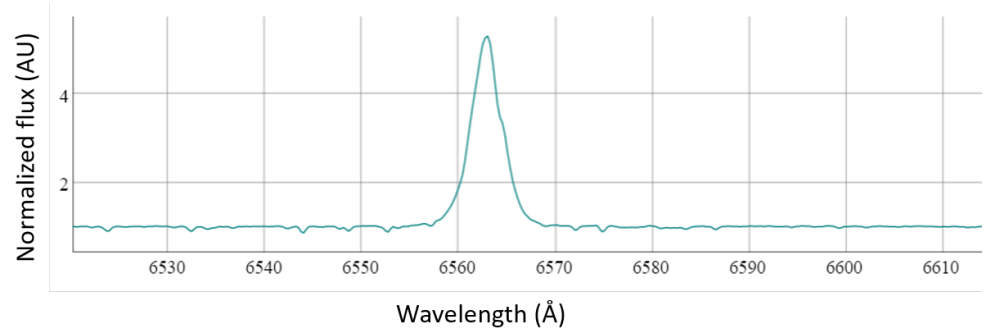

Figure 2.3: Example of an emission peak from Ondřejov archive

lines, corresponding to transitions n=3 to n=2, n=4 to n=2, and so on [14, 22]. The emission character of the Balmer series lines in the spectrum of Be stars is unusual, as absorption hydrogen lines (Fig. 2.2) are much more common. Absorption lines may be explained if we think of stars as hot objects surrounded by a "cool" atmosphere of absorbing gas, most commonly hydrogen gas [10]. If the star surface is too hot, most of the hydrogen atoms get ionized and there is no absorption. Similarly, if the surface is relatively "cold", the light does not have sufficient energy to cause electron transition from n=2 level to higher levels and there are no absorption lines in the Balmer series region either. Thus, absorption Balmer series hydrogen lines are the most common in stars with surface temperature about 9 000 K (spectral type A, weaker in stars of spectral type B and F) [23].

Emission hydrogen lines (Fig. 2.3) are typically seen in the spectra of spiral and irregular galaxies, active galactic nuclei, H II regions (interstellar regions with partially ionized hydrogen gas), and planetary nebulae. In Be stars, the emission lines are supposed to originate from a hot nebulous ring revolving around the star. The ring is formed by matter ejected from the

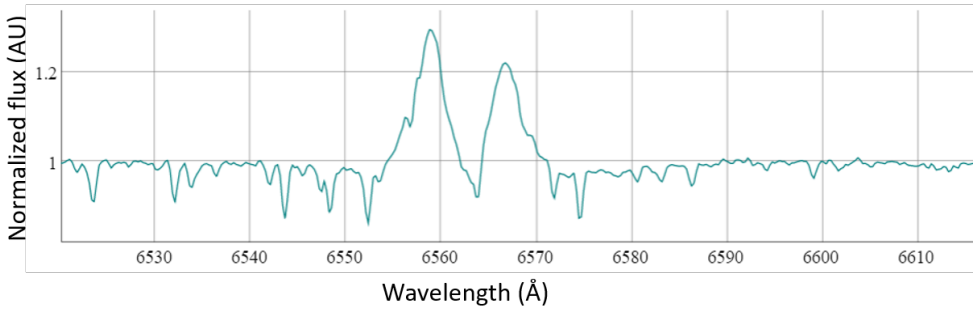

Figure 2.4: Example of a double emission peak from Ondřejov archive

rapidly rotating Be star equator region. The orientation of the Be star axis with respect to the observer is then responsible for the observed wide range of widths of these emission lines (Fig 2.4) [24].

Be stars have long been subject of studies at ASU CAS [25] and Ondřejov archive contains approximately 13 000 spectra with Hα spectral lines [26].

2.1.4 Sources of Spectroscopic Data

Two main data sources of spectroscopic data are used in this study:

- Ondřejov archive available at ASU CAS Data Center [26] storing more than 17 000 spectra, of which approximately 13 000 contain the Hα line. The spectra were acquired using the Perek 2 m telescope, a 700 mm camera with CCD detection, and a Coudé spectrograph [27].

- LAMOST (Large Sky Area Multi-Object Fiber Spectroscopic Telescope) archive presently containing about 10 million spectra (data release 6 version 1 - DR6v1) [28]. LAMOST is a reflective Schmidt telescope with active optics located in the Hebei province southwest of Beijing. The focal surface is circular with a diameter of 1.75 m and is tiled with 4000 fibre-positioning units, each feeding an optical fibre which transfers light to one of sixteen 250-channel spectrographs. Each spectrograph has two 4k x 4k charge-coupled device (CCD) cameras with a 'blue' (370–590 nm) and a 'red' (570–900 nm) channel. It is able to collect light from faint celestial objects down to 20.5 magnitude [29]. The main scientific goals include the LAMOST Extra-galactic Spectroscopic Survey (LEGAS) to explore the large-scale structure of the Universe and the LAMOST Experiment for Galactic Understanding and Exploration (LEGUE) to study stellar spectra, particularly spectra of metal-poor stars in the halo of our galaxy [29].

2.1.5 Spectral Data Storage Formats

The data are typically stored either in Flexible Image Transport System (FITS) [30] or VOTable [31] format. Multiple spectra may be stored in comma-separated-value (CSV) or hierarchical data format (HDF) files [32].

FITS format was designed specifically for astronomical data. The first standard appeared in 1981 [33], the most recent version (4.0) was standardized in 2016 [34]. FITS files contain 1 primary Header/Data Unit (HDU) segment and any number of additional HDUs called extensions. Each segment contains metadata stored in an ASCII header and an optional data unit that may contain data stored either as integers (unsigned 8-bit or signed 16- or 32-bit) or single (32-bit) or double (64-bit) precision floating point real numbers [34]. Typically, the primary data unit contains 1D or 2D image, or 3D data. Standard extensions may be either image or ASCII or binary table extensions [30].

VOTable is an XML-based format recommended by the International Virtual Observatory Alliance [35]. VOTable data may be used in the VO-CLOUD environment, as well.

2.1.6 Spectral Data Preprocessing

As Ondřejov and LAMOST spectrographs have different characteristics, it was necessary to preprocess Ondřejov data in order to adjust their format to that of the LAMOST data [3].

First, wavelength conversion from air (Ondřejov) to vacuum (LAMOST) was applied [36]:

$$\lambda_v = n\lambda_a = (1 + 8.34254.10^{-5} + \frac{2.406147.10^{-2}}{130 - s^2} + \frac{1.5998.10^{-4}}{38.9 - s^2})\lambda_a, \quad (2.2)$$

where $s = \frac{10^4}{\lambda_a}$.

Second, as spectral resolution of the Ondřejov dataset is higher, Gaussian blur and regridding to 140 points (as in LAMOST) was applied to Ondřejov data.

Finally, scaling to zero mean and unit variance was applied to all spectra. For details see [3].

2.2 Active Learning

Machine learning tasks may be classified into 3 basic classes: supervised, unsupervised, and semi-supervised [37, 38]. In supervised learning, the whole training data set is labelled by a teacher and is represented by paired input and desired output values. The agent observes these example input–output pairs and learns a function that maps the input to the output [37]. In unsupervised learning, there is no labelling of the input data and the observations are

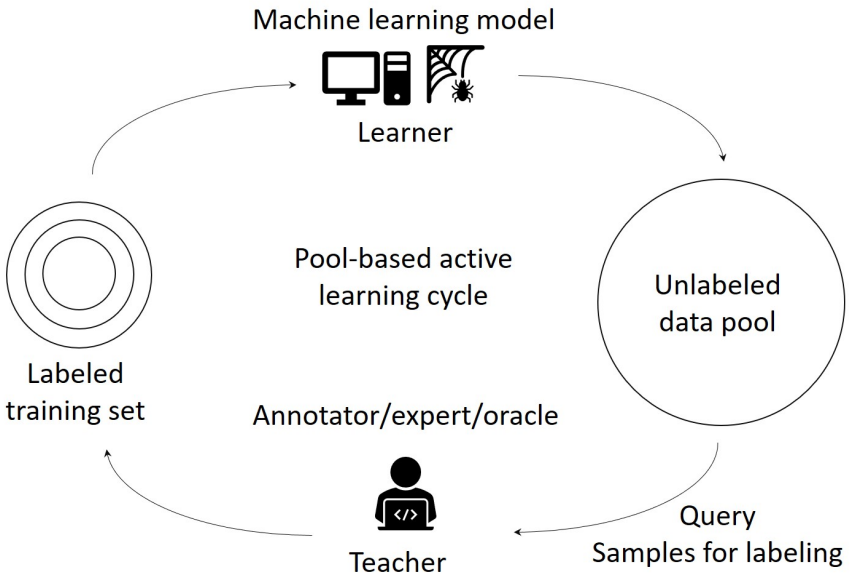

Figure 2.5: Active learning occurs in iterative cycles

clustered based on their mutual similarities [39]. Semi-supervised approaches fall between the previous two, where typically only a small fraction of the training set is labelled by a teacher. Semi-supervised approaches are useful in situations when relatively large amounts of unlabelled data are available and manual labelling of all the data would be too time consuming [40].

Active learning represents a subcategory of semi-supervised machine learning where the learning algorithm in each iteration asks the user/teacher to label a subset of sample data until convergence criteria are met (see Fig. 2.5) [41].

The basic concept of active learning algorithms is very nicely described in [41]: *"The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabelled data instances to be labelled by an oracle (e.g., a human annotator)(see Fig. 2.6). Active learning is well-motivated in many modern machine learning problems, where unlabelled data may be abundant or easily obtained, but labels are difficult, time-consuming, or expensive to obtain."*

2.2.1 Active Learning Scenarios

Three types of active learning scenarios may be distinguished [41]:

- **query synthesis**, where the learner (in our case the neural network

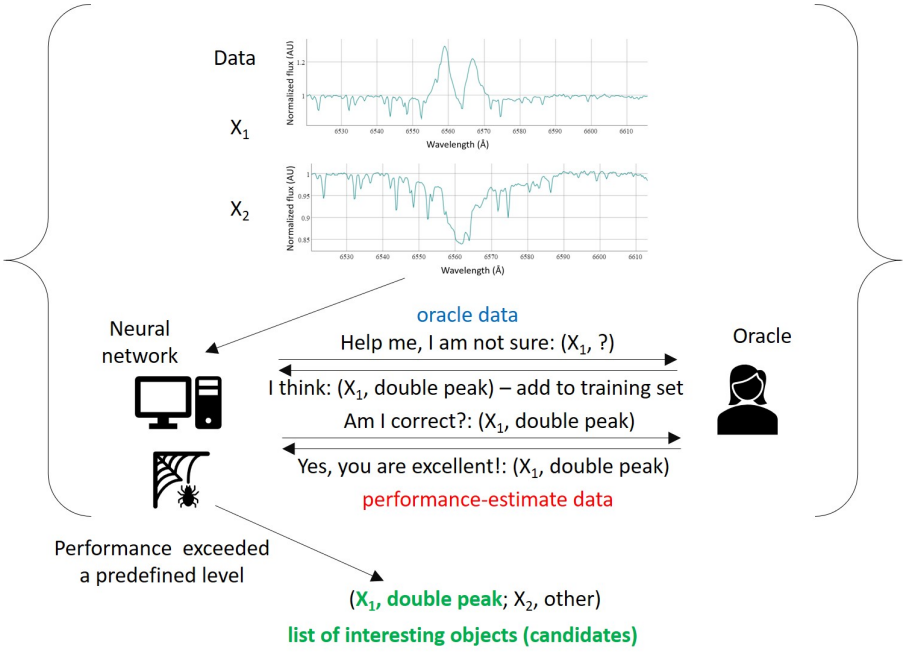

Figure 2.6: Interaction between learner and teacher in active learning

classifier) constructs examples for labelling

- **selective sampling**, where unlabelled data come as a stream and the learner decides whether to query the teacher
- **pool-based active learning**, where the learner chooses from a pool for labelling.

In our case, **pool-based active learning** is applied.

2.2.2 Uncertainty Sampling

How to decide which sample to present for labelling? The most commonly used method is uncertainty sampling, i.e. present the sample where the learner is most uncertain. There are several measures that may be used to estimate the level of uncertainty [41].

The **least confidence (LC)** measure ϕ_{LC} [42] is the simplest:

$$\phi_{LC}(x) = 1 - P_\theta(y^*|x).$$

Here x represents a data sample, y^* is the label with the highest probability $P_\theta(y^*|x)$ under the model θ. The most informative sample – the most promising candidate to be presented for labelling is then:

$$x_{LC}{}^* = \underset{x}{\operatorname{argmax}}(1 - P_\theta(y^*|x)),$$

where $y^* = \underset{x}{\operatorname{argmax}} P_\theta(y|x)$ is again the class label with the highest posterior

probability $P_\theta(y|x)$ under the model θ. The $\underset{x}{\text{argmax}}(f(x))$ function selects the argument x where function $f(x)$ has its maximum. Similarly, the $\underset{x}{\text{argmin}}(f(x))$ function selects the argument x where function $f(x)$ has its minimum.

The **smallest margin (SM)** measure [43] selects samples with the smallest difference between the most probable and the second most probable labels:

$\phi_{SM}(x) = P_\theta({y_1}^*|x) - P_\theta({y_2}^*|x)$ with the most promising candidate:

${x_{SM}}^* = \underset{x}{\text{argmin}}(P_\theta({y_1}^*|x) - P_\theta({y_2}^*|x))$.

Our approach uses the **entropy measure (ENT)** [44]:

$\phi_{ENT}(x) = -\sum_y P_\theta(y|x) log_2 P_\theta(y|x)$ with the most promising candidate:

${x_{ENT}}^* = \underset{x}{\text{argmax}}(-\sum_y P_\theta(y_i|x) log_2 P_\theta(y_i|x))$.

2.2.3 Query by Committee

Another way how to decide which sample to present for labelling is called query-by-committee [45]. The idea is to use a committee $\theta^{(1)}, ..., \theta^{(C)}$ of models trained on the current labelled set. Committee members vote on the individual labellings. The most informative sample is the one about which the committee disagrees the most. The two most commonly used measures of the level of committee disagreement are the vote entropy and the average Kullback-Leibler divergence [41].

In case of **vote entropy (VE)** [44] the most promising candidate for labelling is: ${x_{VE}}^* = \underset{x}{\text{argmax}}(-\sum_i \frac{V(y_i)}{C} log_2 \frac{V(y_i)}{C})$,

where $V(y_i)$ is the number of "votes" for label y_i and C is the number of committee members.

The **Kullback-Leibler divergence** is a measure of the difference between two probability distributions. The most informative sample is the one with the largest average difference between the label distributions of any committee member and the consensus [41].

2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a class of deep feedforward neural networks. Deep means that the networks contain multiple layers between the input and the output layers, feedforward means that information flow is unidirectional and that there are no cycles involved. In contrast to other deep feedforward networks, convolutional networks contain layers that perform convolution (Fig. 2.7), or more precisely sliding dot matrix multiplication. Due to their translation invariance and better generalization capabilities in comparison with fully connected networks, convolutional networks are very popular in the field of image analysis and problems from many other disciplines are often reformulated in form of images and later classified by CNNs. Detailed description of the CNN used in our platform is included in the results section.

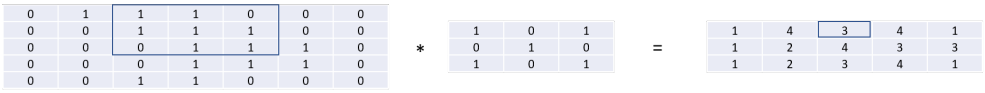

Figure 2.7: Matrix convolution operation performed in convolutional neural networks

2.4 VO-CLOUD

VO-CLOUD is a web-based platform developed at the Faculty of Information Technology, CTU in Prague, in collaboration with the Astronomical Institute of the Czech Academy of Sciences (ASU) in Ondřejov. Its architecture was inspired by concepts used by the International Virtual Observatory Alliance (IVOA), mainly by the Universal Worker Service (UWS) RESTful pattern [46, 47]. Its main purpose is exploratory analysis of astronomical data, in particular analysis of astronomical object spectra using machine learning approaches. Its design is optimally suited for performing complex time and computational power consuming operations on big data stored remotely using simple JSON configurations and nondemanding interactive computation output visualization on a local computer or even on a cell phone.

The core of the system is the Java-based VO-CLOUD master server that is presently deployed on the `betelgeuse` computer of the ASU using the WildFly application server. It provides web interface for VO-CLOUD users allowing them to formulate their requests, start computational tasks ("jobs") and analyze obtained results. Jobs get delegated to so called "workers", software engines responsible for carrying out particular types of tasks. The master server also communicates with a PostgreSQL database where user accounts, history of previous experiments, and the list of available workers are stored. Fig. 2.8 shows the deployment diagram - modification of the deployment diagram from [2].

2.4.1 Typical VO-CLOUD Workflow

Starting an analysis in VO-CLOUD is simple.

1. Login at: `https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse`. In case this is the first time you use the platform, you will have to fill the registration form to obtain your username and password.

2. Upload the data to be analyzed by selecting the *Manage filesystem* option in the main menu bar. Place the data into your subfolder of the *DATA* folder. In case you wish to reanalyze data that you already uploaded before, you may skip this point.

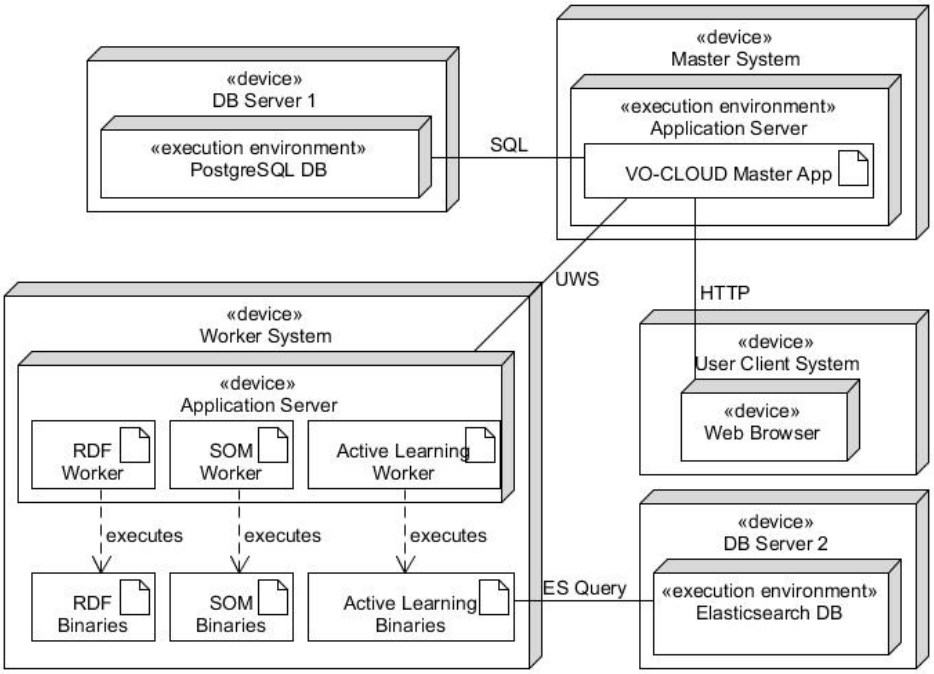

Figure 2.8: VO-CLOUD deployment diagram modified from [2]

3. Use the *Create new job* in the main menu bar to select the desired task (worker).

4. Create a JSON configuration file defining all necessary parameters of your requested task. You may either write or upload your own JSON script or select a precreated configuration.

5. *Save and run* your task.

6. Review the obtained results by pressing the *Details* button that appears next to your job in the job list section (*Jobs* option in the main menu bar).

Chapter 3

Realization

To provide active learning capabilities, the VO-CLOUD system was modified at various levels and a new active learning module was added. VO-CLOUD system is available at `https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse`.

The implementation of the active learning module uses a combination of several technologies:

- Java EE, Java Server Faces (JSF), Enterprise Java Beans (EJB) and WildFly application server for VO-CLOUD master server web interface
- Python and its specialized libraries for the implementation of the active learning algorithm
- JavaScript for interactive web page design
- Elasticsearch database system for storing metadata and results of the active learning process.

Following sections describe the individual necessary modifications in detail.

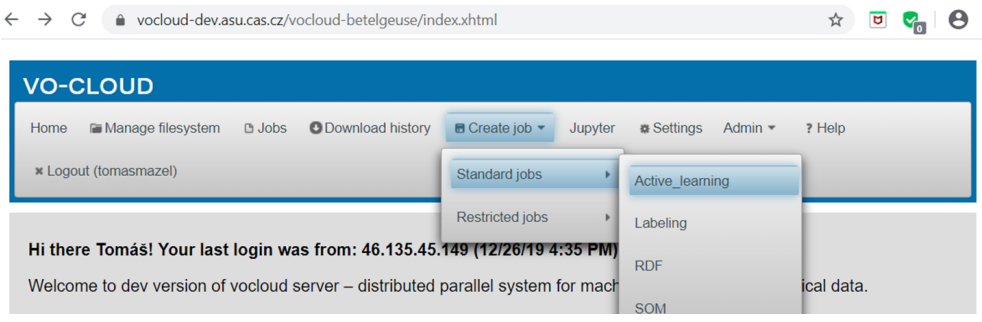

Figure 3.1: Starting active learning job in VO-CLOUD environment

3.1 VO-CLOUD Web Interface

3.1.1 Adding Active Learning Task Option to VO-CLOUD

The active learning option has been added to the VO-CLOUD web interface menus. This step does not require any underlying software modifications and can be done directly within the VO-CLOUD environment. With administrator privileges it is possible to add new workers and define their properties using the Admin option in the main VO-CLOUD menu. However, at this point the system is not able to run the active learning job as it lacks the definition of the interface with the software used to implement the worker.

For this purpose, the XML configuration file describing the individual workers had to be modified. Active learning option and its settings were added and the Universal Worker System (UWS) was recompiled and deployed to VO-CLOUD using the WildFly application server running at the `betelgeuse` computer of the Astronomy Institute. This finally allowed VO-CLOUD and VO-CLOUD users to communicate with the Python-based active learning software.

The `uws-config.vocloud-betelgeuse.xml` configuration file adapted from[1]:

```
<?xml version="1.0" encoding="utf-8"?>
<ns:uws-settings
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:ns='http://vocloud.ivoa.cz/universal/schema'
   xsi:schemaLocation='http://vocloud.ivoa.cz/universal/schema
configSchema.xsd'>
   <ns:vocloud-server-address>http://localhost:8080/vocloud-be-
telgeuse</
ns:vocloud-server-address>
   <ns:local-address>http://localhost:8080</ns:local-address>
   <ns:max-jobs>2</ns:max-jobs>
   <ns:description>Universal UWS worker</ns:description>
   <ns:default-execution-duration>3600</ns:default-execution-
duration>
   <ns:max-execution-duration>3600</ns:max-execution-duration>
      <ns:workers>
         <ns:worker>
            <ns:identifier>active_learning</ns:identifier>
            <ns:description>Active_learning</ns:description>
            <ns:restricted>true</ns:restricted>
            <ns:binaries-location>/usr/local/workers/active
_learning</ns:  binaries-location>
            <ns:exec-command>
```

```
               <ns:command>python3</ns:command>
               <ns:command>${binaries-location}/run-active-
learning.py
</ns:  command>
               <ns:command>${config-file}</ns:command>
            </ns:exec-command>
         </ns:worker>
         <ns:worker>
            <ns:identifier>preprocessing</ns:identifier>
            <ns:description>Preprocessing</ns:description>
            <ns:restricted>true</ns:restricted>
            <ns:binaries-location>/usr/local/workers/preprocess-
ing<
/ns:binaries-location>
            <ns:exec-command>
               <ns:command>python3</ns:command>
               <ns:command>${binaries-location}/run_preprocess-
ing.py
</ns:command>
               <ns:command>${config-file}</ns:command>
            </ns:exec-command>
         </ns:worker>
         <ns:worker>
            <ns:identifier>som</ns:identifier>
            <ns:description>SOM</ns:description>
            <ns:restricted>false</ns:restricted>
            <ns:binaries-location>/usr/local/workers/som</ns:bi-
naries-
location>
            <ns:exec-command>
               <ns:command>python3</ns:command>
               <ns:command>${binaries-location}/run.py</ns:com-
mand>
            </ns:exec-command>
         </ns:worker>
         <ns:worker>
            <ns:identifier>rdf</ns:identifier>
            <ns:description>RDF</ns:description>
            <ns:restricted>false</ns:restricted>
            <ns:binaries-location>/usr/local/workers/rdf</ns:bi-
naries-loca-tion>
            <ns:exec-command>
               <ns:command>python3</ns:command>
               <ns:command>${binaries-location}/runRF.py</ns:com-
```

```
mand>
                <ns:command>${config-file}</ns:command>
            </ns:exec-command>
        </ns:worker>
    </ns:workers>
</ns:uws-settings>
```

The Java code for VO-CLOUD was downloaded from `https://github.com/vodev/vocloud` to `/home/mazeltom/vocloud` folder at the `betelgeuse` server at ASU CAS and compiled using the `mvn package -Pvocloud-betelgeuse` command in the appropriate folder. For deployment, the `ssh -L 9900:localhost:9900 betelgeuse` command was used for remote login as root and the `WildFly JBoss` utility was used for deployment.

3.1.2 Starting New Active Learning Job

VO-CLOUD server GUI interface was modified to allow users to select desired active learning parameters when starting an active learning job. An active learning job in VO-CLOUD may be started in the following way.

1. Use the *Create new job* in the main menu bar to select the desired task (worker)(Fig. 3.1).

2. Create a JSON configuration file defining all necessary parameters of your requested task. You may either write or upload your own JSON script or select a precreated configuration (Figs. 3.2, 3.3, 3.4).

3. *Save and run* your task (Fig. 3.4).

Another substantial modification was the inclusion of a web page displaying the results together with the visualization of the spectra of selected objects (Figs. 3.5, 3.6, 3.7, 3.8). An important additional functionality is the display of metadata associated with the spectra. Importantly, users can manually reclassify the objects and store labels and comments in the underlying Elasticsearch database. All these additions were done by modifying the underlying XHTML template and the JavaScript code included in it.

When an active learning job is started, the XHTML template (file named `spectra_list_html.template`) is "filled" by the `run-active-learning.py` Python script, i.e. the placeholders in the XHTML template get replaced by data supplied by the Python script. For example, all instances of `${md}` in the JavaScript are substituted by metadata generated in Python. The replacement is performed by the `html_code = html_template.substitute`({"list":"".join (spectra_list), "labels2add_fname": labels2add_fname, "spectra2add_fname": spectra2add_fname, "md": metadata, "comments": comment_list, "cats": cat-

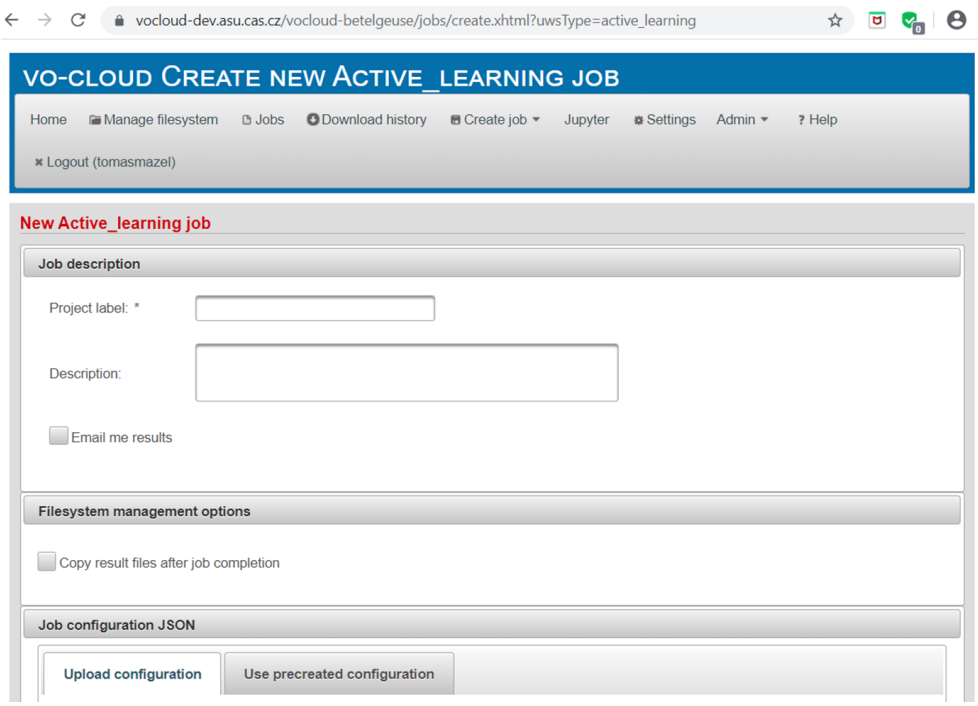

Figure 3.2: Specifying new active learning job

egories_str, "fname": fname_list, "prediction": prediction_list, "cat": cat_list, "lab": label_list, "oracle_perf": oracle_perfest_list, "random_sample_size": random_sample_size, "mdcols" : metadata2show})" command in the `_generate_spectra` function.

3.1.3 Displaying Results and Labelling

Figures 3.5, 3.6, 3.7, 3.8 show the typical workflow in displaying and labelling the results of a neural network iteration and available functions for displaying spectra and metadata of samples preselected for labelling. The typical order of steps is the following:

1. In the *Jobs* window go to the *Details*. This will open a window displaying the list of spectra to be labelled. The user may view both the preprocessed and the original spectra (see Fig. 3.9), as well as their corresponding metadata.

2. Start labelling the individual samples by clicking on the radio button with the selected label or by using the corresponding keyboard shortcut (*Alt + number*).

3. Label all the presented samples. In case you want to change the previ-

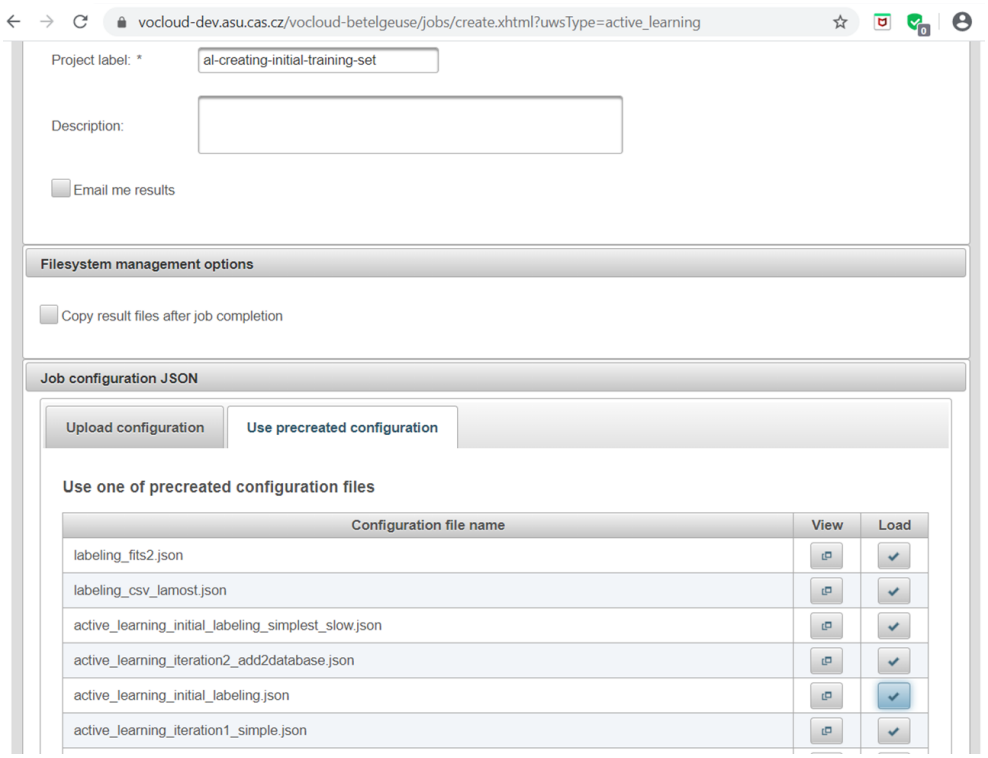

Figure 3.3: Selecting precreated configuration

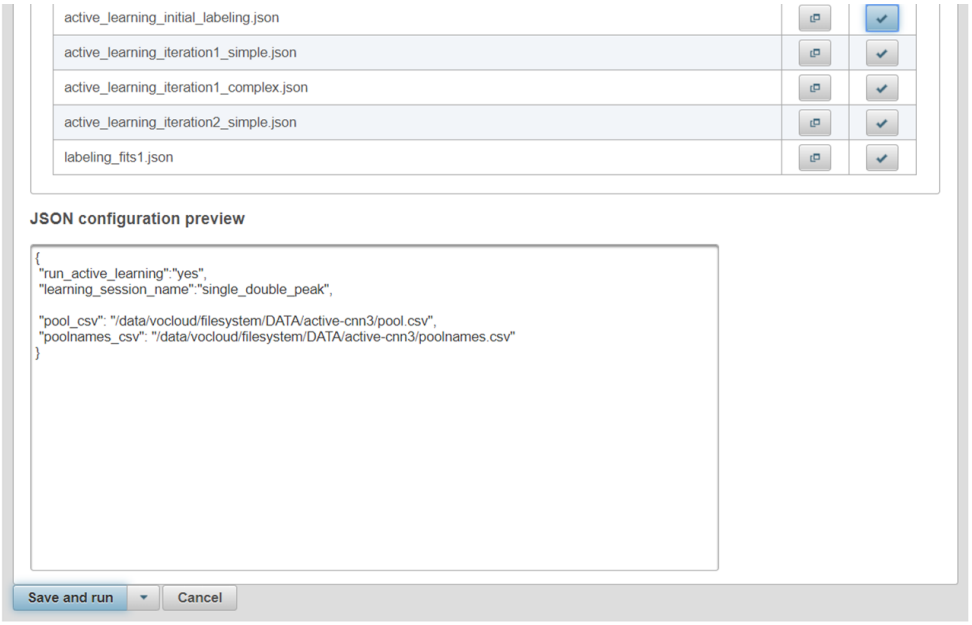

Figure 3.4: Simple JSON configuration to create initial training set

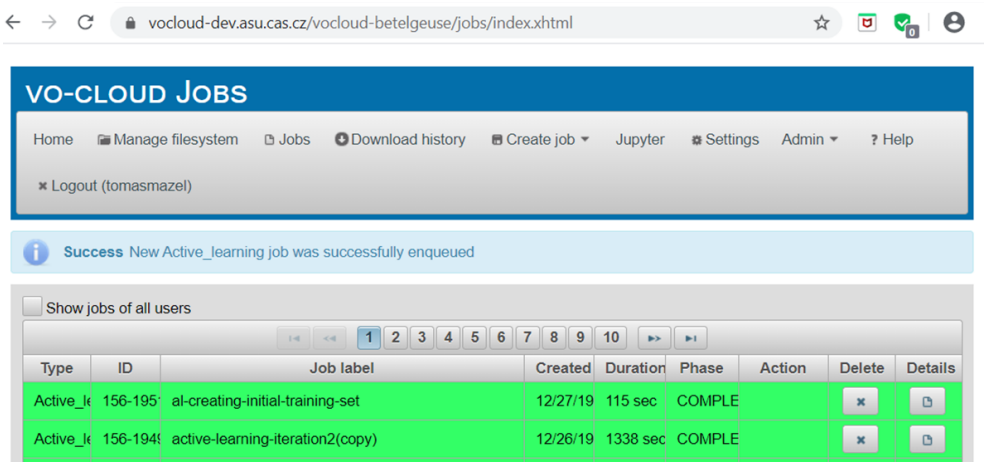

Figure 3.5: List of jobs - going to details

ous label, you may use the *Previous* button or you may simply click on the name of the spectra.

4. You may review all your labellings together with the metadata using the *All metadata* button. You may also view the individual or multiple spectra at any time, you may also use the available zoom in function.

5. When you are satisfied with the labels, you may save them by pressing the *Save* button and download the CSV file containing spectra IDs together with the corresponding labels and comments *Download labels* and the CSV file containing the labels together with the processed spectra (*Download labelled spectra*). These files may then be uploaded to the folder of your choice in the VO-CLOUD filesystem and added in the next iteration to the Elasticsearch database (`"labels2database csv"` in the JSON configuration file) and to the training set (`"training set addition csv"` in the JSON configuration file).

6. You may view the performance estimate, i.e. the percentage of samples from the performance-estimate set where your label equalled the neural network prediction by pressing the *Performance* button. This will also allow you to see the overall statistics, i.e. the number of samples in individual categories.

7. Start a new iteration using the configuration stored in the `new_config.json` file.

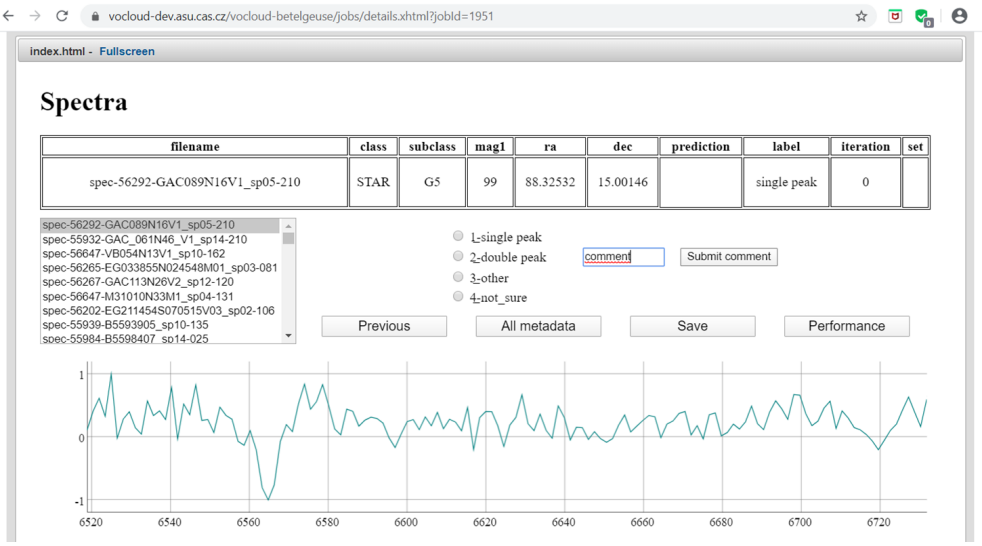

Figure 3.6: List on the left allows to select a spectrum for evaluation. Table on top includes preselected metadata. Panel in the middle allows to classify the visualized spectra and add comments.

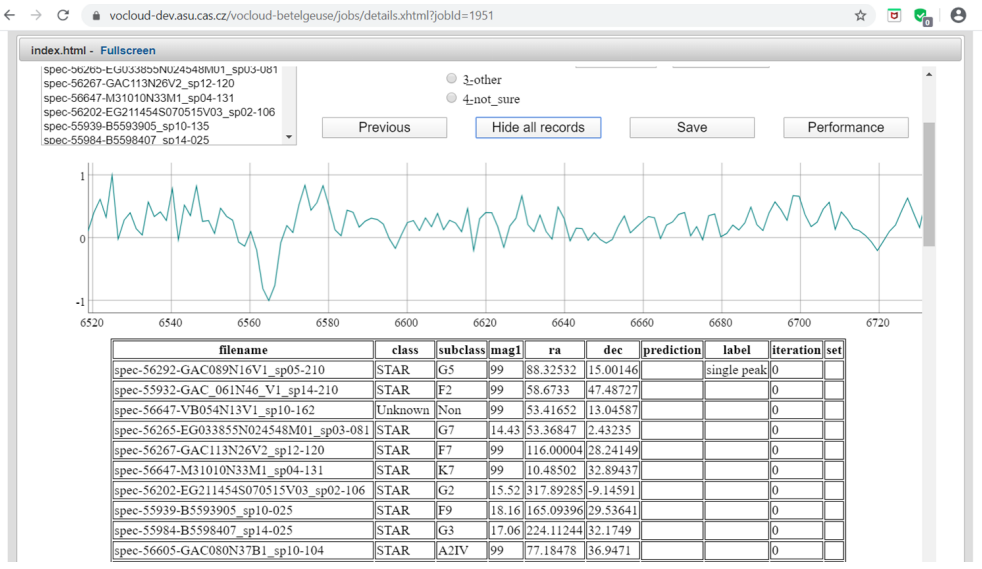

Figure 3.7: All metadata button provides an option to see metadata of all preselected records

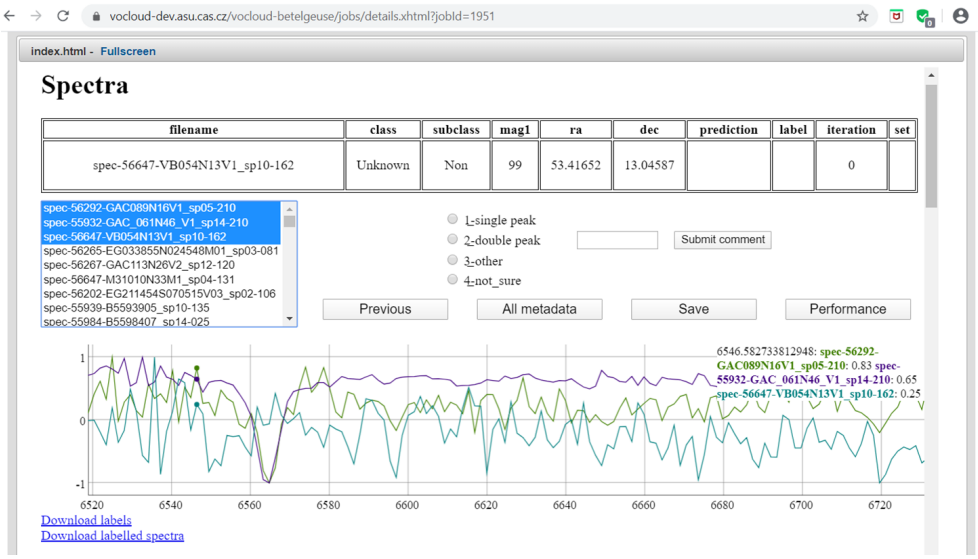

Figure 3.8: Viewing multiple spectra

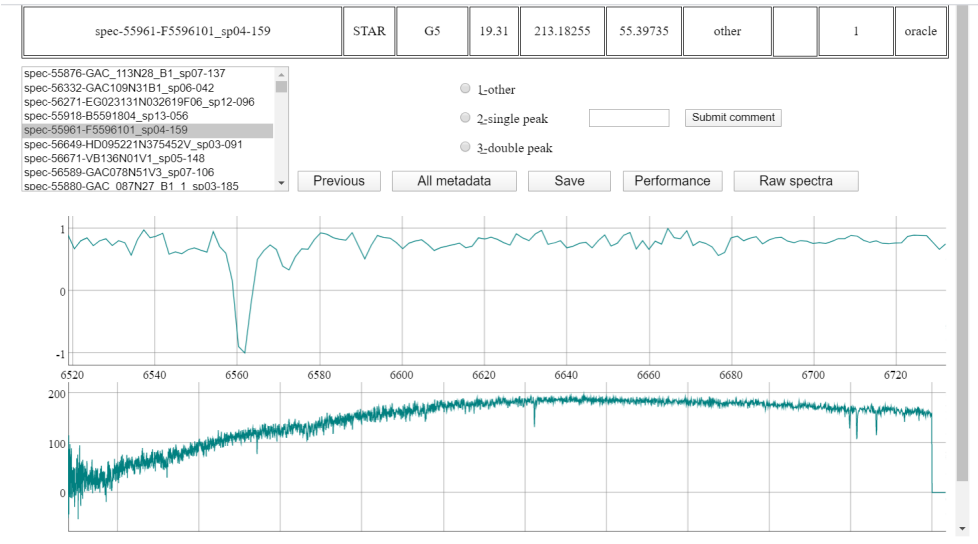

Figure 3.9: Viewing both preprocessed and original spectra

3.2 Active Learning Worker

The actual implementation of the active learning worker was done in Python. The main `run-active-learning.py` file referenced in the UWS XML configuration file contains the main executable, the `run_preprocessing` and the `_generate_spectra` functions. The `run_preprocessing` function parses the JSON configuration input file and executes functions corresponding to the selected JSON options. The `_generate_spectra` function prepares the list of spectra along with the corresponding metadata and fills the xhtml template. The naming of the functions and the overall structure of the code was designed to be similar to the Python codes performing other types of jobs available in the VO-CLOUD system, especially preprocessing.

3.2.1 Parsing Input JSON Configuration File

The JSON configuration is parsed by the `run_preprocessing` function contained in the `run-active-learning.py` file.

The main settings optionally defined in the JSON configuration file are the following:

Input:
- `learning_session_name`: optional project name
- `iteration_num`: 0 - initial training set preparation, 1.. - individual iterations
- `training_set_csv`: CSV file containing IDs and labels of spectra in the training set (if not specified, the labelling-only regime is triggered)
- `training_set_addition_csv`: CSV file containing IDs and labels of spectra to be added to the training set (typically created in previous iteration)
- `pool_csv`: CSV file containing all preprocessed spectra to be analyzed (in our case the LAMOST DR2 database containing about 4 million spectra)
- `poolnames_csv`: CSV file containing IDs of all preprocessed spectra to be analyzed (presently significantly improves the speed)
- `labels2add_csv`: CSV file containing IDs, labels and comments of spectra evaluated in previous iteration

Classification:
- `categories`: classes of spectra (e.g. [”single peak”, ”double peak”, ”other”])

Output:
- `batch_size`: number of spectra to be analyzed
- `random_sample_size`: number of randomly selected spectra to be analyzed, default: 0
- `oracle_csv`: CSV file containing IDs of spectra to be analyzed by the ”oracle” (expert) - typically spectra where the active learning algorithm finds the

highest degree of uncertainty, default: ”oracle.csv”
- `performance_estimation_csv`: CSV file containing IDs of spectra randomly selected from the pool(names), default: ”perf-est.csv”

Metadata display:
- `metadata2show`: list of metadata items to be displayed, e.g. [”filename”, ”class”, ”subclass”, ”mag1”, ”ra”, ”dec”, ”prediction”, ”label”, ”iteration”, ”set”] - used by default

3.2.2 Examples of JSON configuration files:

Iteration 0 - preparing the initial training set:

```
{
   "run_active_learning":"yes",
   "learning_session_name":"single_double_peak",
   "pool_csv":"/data/vocloud/filesystem/DATA/active-learning/
pool.csv"
}
```

Iteration 0 - preparing the initial training set (faster version):

```
{
   "run_active_learning":"yes",
   "learning_session_name":"single_double_peak",
   "pool_csv":"/data/vocloud/filesystem/DATA/active-learning/
pool.csv",
   "poolnames_csv":  "/data/vocloud/filesystem/DATA/active-learn-
ing/poolnames.csv"
}
```

Iteration 1 - active learning (simple):

```
{
   "run_active_learning":  "yes",
   "learning_session_name":"single_double_peak",
   "training_set_csv":"/data/vocloud/filesystem/DATA/active-learn-
ing/training-set.csv",
   "iteration_num":  1,

   "pool_csv":"/data/vocloud/filesystem/DATA/active-learning/
pool.csv",
   "poolnames_csv":"/data/vocloud/filesystem/DATA/active-learning/
poolnames.csv",

   "random_sample_size":  30,
```

```
   "batch_size":  130
}
```

Iteration 1 - active learning (complex):

```
{
   "run_active_learning":  "yes",
   "learning_session_name":"single_peak_double_peak",
   "training_set_csv":"/data/vocloud/filesystem/DATA/active-learn-
ing/training-set.csv",
   "pool_csv":"/data/vocloud/filesystem/DATA/active-learning/
pool.csv",
   "metadata2show":["filename","class","subclass","mag1","ra",
"dec","prediction","label","iteration","set"],

   "training_set_addition_csv":  "",
   "iteration_num":  1,

   "csv_spectra_file_names":"/data/vocloud/filesystem/DATA/active-
learning/poolnames.csv",
   "csv_spectra_file":"/data/vocloud/filesystem/DATA/active-learn-
ing/pool.csv",
   "performance_estimation_csv":"/data/vocloud/filesystem/DATA/ac-
tive-learning/perf-est.csv",
   "oracle_csv":"/data/vocloud/filesystem/DATA/active-learning/or-
acle.csv",
   "csv_spectra_file2":"csv_spectra_file2.csv",

   "random_sample_size":  30,
   "batch_size":  130
}
```

Iteration 2 - active learning with labels from previous iteration being stored in the Elasticsearch database:

```
{
   "run_active_learning":  "yes",
   "learning_session_name":"single_double_peak",
   "training_set_csv":"/data/vocloud/filesystem/DATA/active-learn-
ing/training-set.csv",

   "pool_csv":"/data/vocloud/filesystem/DATA/active-learning/pool.
csv",
   "poolnames_csv":"/data/vocloud/filesystem/DATA/active-learning
/poolnames.csv",
   "metadata2show":  ["filename","class","subclass","mag1","ra",
```

```
"dec","prediction","label","iteration","set"],

   "training_set_addition_csv":"/data/vocloud/filesystem/DATA/ac-
tive-learning/spectra2add_1.csv",
   "labels2database_csv":  "/data/vocloud/filesystem/DATA/active-
learning/ labels2add_1.csv",
   "iteration_num":  2,

   "performance_estimation_csv":  "/data/vocloud/filesystem/DATA/
active-learning/perf-est.csv",
   "oracle_csv":  "/data/vocloud/filesystem/DATA/active-learning/
oracle.csv",
   "csv_spectra_file2":"csv_spectra_file2.csv",

   "random_sample_size":  30,
   "batch_size":  130
}
```

3.2.3 HTML Page Generation

The HTML page containing the spectra to be analyzed is generated by the `_generate_spectra` function contained in the `run-active-learning.py` file. This function replaces all placeholders in the `spectra_list.html.template` file. Metadata get imported either from the Elasticsearch database, from a CSV file or from original FITS files. The rest of the input comes from the parsed JSON input file.

3.2.4 Required Python Libraries

Several non-standard Python libraries are required and had to be installed, in particular:

- `astropy` handling astronomy data [48]
- `pandas` data analysis [49]
- `keras` neural network interface [50, 51]
- `tensoflow-gpu` interface with `TensorFlow` [52]
- `matplotlib` data visualization [53]
- `numpy` high-level mathematical functions [54]
- `scipy` scientific computing [55]
- `scikit-learn` free machine learning library [56]
- `elasticsearch` communication with Elasticsearch database [57]

3.2.5 Data handling Support Functions

Most data handling support functions are defined in the `data_handler.py` file. It includes functions that enable reading spectra and metadata from

both CSV and FITS files. Most functions were modified or taken directly from the `data_handler.py` file of the preprocessing job [2].

The `load_set_al` function is used to extract spectral data for samples the names of which are contained the supplied list. The function produces files containing the preprocessed spectra used by the neural network classification, as well as the original ("raw") spectra. The `load_metadata_set` function allows to import required metadata from CSV files.

The `__spectra_rebinning` and `_normalize` functions are applied when original FITS files are used for labelling.

3.2.6 Active Learning Core

Individual active learning iterations are performed by functions defined in the `activecnn.py` file prepared together with its support functions by Ondřej Podsztavek in frames of his bachelor and master thesis [3].

The support functions are included in the files contained in the `active_cnn` folder. The model is defined in the `get_model` function in the `model.py` file. The structure of the network and the information flow is depicted in Figure 3.10. The input has 140 channels corresponding to normalized values of binned spectrum intensity in wavelength region between 6519 and 6732. The network then involves 3 cycles of 1D convolution (individual cycles containing 2 layers with 64, 128 and 256 3-pixel filters with rectified linear unit activation function and 2-pixel `maxpooling`) followed by 2 fully connected layers with 512 units each and with a dropout probability of 0.5 (Fig. 3.11). The output layer has 3 channels with `softmax` activation corresponding to three classification categories (single peak, double peak and other) [3].

The convolutional layers are used for feature extraction. In each of the three convolutional cycles two convolutions are followed by `maxpooling` with pool size 2, stride 2 (i.e. non-overlapping regions) and no padding. The role of maxpooling is to reduce dimensionality, reduce the risk of overfitting and improve translational invariance provided by the convolutional layers. The 2 fully connected layers fulfill the task of mapping the extracted features to the predefined classifier categories. Dropout (randomly disconnected nodes) is applied to reduce the possibility of overfitting.

The active learning scripts are based on `Keras`, a free open-source library written in Python [50, 51]. It may run both on top of `Theano` and `TensorFlow`. `Theano` is a free open-source Python library for mathematical operations, in particular those involving matrix manipulations [58]. `TensorFlow` is a free open-source Python library developed specially for machine learning application [52]. Both `Theano` and `TensorFlow` can be efficiently run on multiple CPU or GPU units.

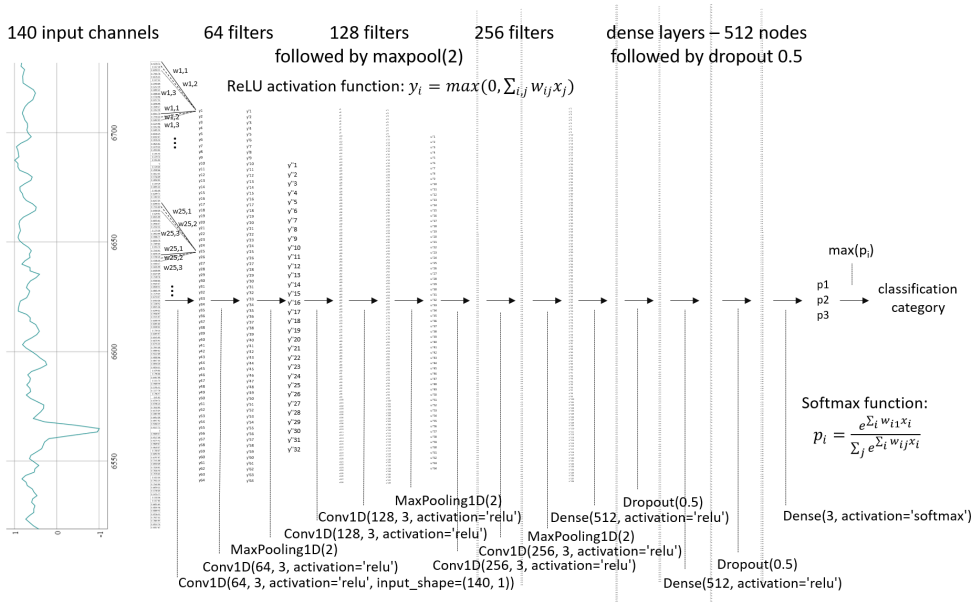

Figure 3.10: Convolutional network workflow [3]

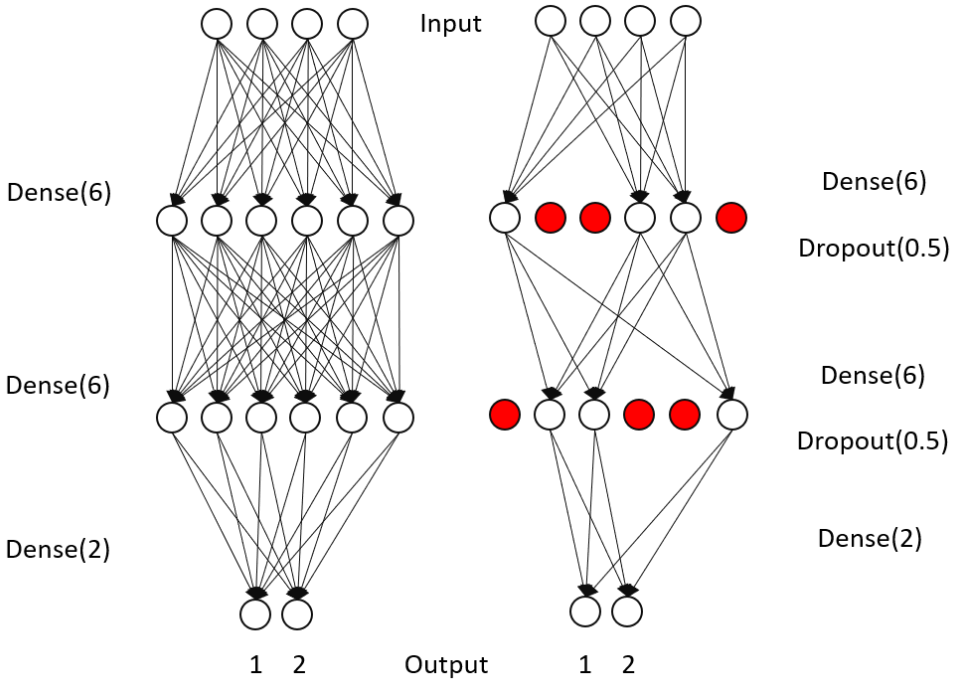

Figure 3.11: Final dense layers with 0.5 droupout. The real network uses Dense(512), Dense(512) and Dense(3) [3].

3.2.7 Network Training

The original training set was created based on data from the Ondřejov library. Briefly, the Ondřejov dataset was split into training, validation and test set [3]. The training set comprising approximately 70 % of the whole dataset was used for optimization of the individual weight functions, the validation set representing around 20 % of the dataset was used for hyperparameter and architecture optimization, and the remaining 10 % data were used for testing for neural network evaluation [3].

As most of the spectra belong to the absorption class and the other two categories (single peak emission spectra and double peaks) are relatively rare, SMOTE balancing was applied [3]. SMOTE (Synthetic Minority Oversampling Technique) aims to improve the distribution balance by artificially creating additional underrepresented samples [59]. SMOTE can be imported from the imblearn Python library [3].

The categorical cross-entropy loss function is used as network optimization function. In general, the cross-entropy (CE) function is defined as [60, 61]

$$CE = -\sum_{i}^{C} t_i log(f(s_i)) \tag{3.1}$$

where t_i represent the ground truth, s_i are the CNN scores, and f is the activation function. In the case of our convolutional network applying the softmax function in the last step with only one positive class s_p this equation reduces to

$$CE = -log\frac{e^{s_p}}{\sum_{i}^{C} e^{s_i}} \tag{3.2}$$

In each iteration, newly labelled spectra are added to the initial training set. As output, we obtain a list of spectra where the active learning algorithm found the highest cross-entropy values. This list saved in the `oracle.csv` file is then presented to the teacher (oracle) for labelling. In addition, random spectra are selected for performance estimation. Once the percentage of correct predictions, i.e. the CNN predictions matching the labels given by the oracle (the expert teacher), in the performance estimate set exceeds a predefined level, the active learning process may be terminated.

3.3 Elasticsearch Database for Storing Metadata

One of the objectives of the project was to store the results of the active learning classification together with the most important metadata information about the classified objects. For this purpose, the Elasticsearch engine has been selected as it provides very fast full-text search allowing to quickly select suitable candidate spectra for further analysis, as well as an easy and economical way of adding additional columns with spare entries [62].

Elasticsearch is a free open-source full-text search engine with an HTTP JSON interface developed in Java. Its first version was released in 2010 by Shay Banon [62]. Elasticsearch is based on `Apache Lucene` library that allows very fast full text indexing and search and also a fuzzy search based on edit distance. Elasticsearch together with a data collection and log-parsing engine called `Logstash` and an analytics and visualization platform `Kibana` form an integrated solution referred to as the Elastic or `ELK stack` [62].

Elasticsearch has many features that make it ideally suitable for our project. Elasticsearch is [63]:

- NoSQL, non-relational
- open-source
- horizontally scalable, distributable over hardware devices
- very fast at fulltext search
- reliable

Elasticsearch is built for speed [63]. Elasticsearch does not have transactions in the typical sense. There is no way to rollback an already submitted document. Changes are visible after the index gets refreshed, i.e. typically once per second.

Elasticsearch has been installed on the `betelgeuse` server of the Astronomical Institute of the Czech Academy of Sciences.

3.3.1 Elasticsearch Installation

The installation process is relatively simple.

1. Download: `curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.7.0.tar.gz`.

2. Archive extraction: `tar -xvf elasticsearch-6.7.0.tar.gz`

3. Starting the program: `./elasticsearch-6.7.0/bin/elasticsearch`.

3.3.2 Examples of Usage

Index creation:

```
curl -XPUT 'http://localhost:9200/lamost-dr5-v3/'
```

Adding a document:

```
curl -XPUT 'http://localhost:9200/lamost-dr5-v3/1' -d '{
   "doc" :  {
```

```
        "ra":"72.030839",
        "dec" :  "44.839599",
        "prediction" :  "double peak"
     }
  }'
```

Partial update:

```
  curl -XPOST 'http://localhost:9200/lamost-dr5-v3/1/_update'-d'{
     "doc":{
        "label":  "single peak",
        "comment":  "not insteresting"
     }
  }'
```

Retrieving a document (in this case based on its id):

```
  curl -XGET 'http://localhost:9200/lamost-dr5-v3/1'
```

Searching for a document:

```
  curl -XGET http://localhost:9200/candidates-metadata-test1/
_search?q=single
  curl -XGET http://localhost:9200/lamost-dr5-v3/_search?q=file-
name:spec-56298-HD110736N135131B01_sp05-092
```

Deleting records:

```
  curl -XDELETE localhost:9200/candidates-metadata-test1
```

Getting database statistics:

```
  curl -XGET "http://localhost:9200/_cat/shards?v"
```

3.3.3 Elasticsearch.py

`Elasticsearch.py` is a Python library allowing communication with Elasticsearch databases [57].

Installation is simple:

```
  pip install elasticsearch.
```

Establishing connection to database:

```
  from elasticsearch import Elasticsearch
  es=Elasticsearch([{'host':'localhost','port':9200}])
```

Storing data:

```
  doc={'ra':'21.896694','dec':'62.614417','prediction':'double
peak','filename':'a201306110029.fits','timestamp':datetime.now()}
```

```
   es.index(index='ondrejov', doc_type='doc', id='a201306110029',
body = doc)
```

Partial update:

```
   es.update(index='lamost-dr5-v3',doc_type='doc',id='spec-55967-
GAC-_073N44_V4_sp15-074',body={'doc':{'label':"double peak','com-
ment':'interesting' }})
```

Retrieving data:

```
   res=es.get(index='ondrejov',doc_type='doc',id='a201306110029')
   print res['_source']
```

Deleting data:

```
   es.delete(index='ondrejov',doc_type='doc',id='a201306110029')
```

Simple search:

```
   res=es.search(index='lamost-dr5-v3',body={'query':{'match_all':
{}}})
   res= es.search(index='lamost-dr5-v3',body={'query':{}})
```

Match operator - close matches:

```
   res= es.search(index='ondrejov',body={'query':{'match':{'file-
name':  'a201306110029'}}}, size=1)
```

Term operator - exact matches:

```
   res=es.search(index='ondrejov',body='query':'term':'file-
name':'a201306110029',size=1)
```

Bool and filter operator:

```
   res= es.search(index='lamost-dr5-v3',body={
      'query':{
         'bool':{
            'must':{
               'match':{
                  'prediction':'double peak'
               }
            },
            "filter":{
               "range":{
                  "mag1":{
                     "lt":1
                  }
               }
            }
```

```
        }
      }
   })
   print res['hits']['hits']
```

3.3.4 Building Elasticsearch Database

Currently, two separate databases (indices in terms of Elasticsearch language) are available, one, called `ondrejov`, for the Ondřejov dataset and one, labelled `lamost-dr2`, for LAMOST DR2 (data release 2). They were established by importing spectral metadata stored in comma-separated-value files using the `Logstash` tool for parsing log files. This was done directly at the `betelgeuse` computer after logging in with user rights.

As new and new LAMOST data releases get published, additional databases will be added. Presently, data from LAMOST DR5 (data release 5) version 3 are being processed and `lamost-dr5-v3` database will be available soon.

The command used to establish the LAMOST database was:

```
logstash -f metadata-lamost.conf
```

with the following `metadata-lamost.conf` configuration file:

```
input {
   file {
      path=>"/home/mazeltom/workers/active-learning/data/lamost-
dr2-metadata.csv"
      start_position => "beginning"
      sincedb_path => "/dev/null"
   }
}
filter {
   csv {
      separator => ","
      columns => ["filename", "obsid", "designation", "obsdate",
"lmjd","mjd","planid","spid","fiberid","ra_obs","dec_obs","snru",
"snrg","snrr","snri", "snrz", "objtype", "class", "subclass","z",
"z_err", "magtype", "mag1", "mag2", "mag3", "mag4","mag5", "mag6",
"mag7","tsource","fibertype","tfrom","tcomment","offsets", "off-
set_v","ra","dec","fibermask"]
   }
   mutate {convert => ["obsid","integer"] }
   mutate {convert => ["lmjd","integer"] }
   mutate {convert => ["mjd","integer"] }
   mutate {convert => ["fiberid","integer"] }
```

```
   mutate {convert => ["fibermask","integer"] }
   mutate {convert => ["mjd","integer"] }
   mutate {convert => ["spid","integer"] }
   mutate {convert => ["ra_obs","float"] }
   mutate {convert => ["dec_obs","float"] }
   mutate {convert => ["offset_v","float"] }
   mutate {convert => ["mag1","float"] }
   mutate {convert => ["mag2","float"] }
   mutate {convert => ["mag3","float"] }
   mutate {convert => ["mag4","float"] }
   mutate {convert => ["mag5","float"] }
   mutate {convert => ["mag6","float"] }
   mutate {convert => ["mag7","float"] }
   mutate {convert => ["snru","float"] }
   mutate {convert => ["snrg","float"] }
   mutate {convert => ["snrr","float"] }
   mutate {convert => ["snri","float"] }
   mutate {convert => ["snrz","float"] }
   mutate {convert => ["z","float"] }
   mutate {convert => ["z_err","float"] }
   mutate {convert => ["ra","float"] }
   mutate {convert => ["dec","float"] }
}
output {
   elasticsearch {
      hosts => "http://localhost:9200"
      index => "lamost-dr2"
   }
   stdout { codec => rubydebug }
}
```

Similarly, the command used to establish the Ondřejov database was:

```
logstash -f metadata-ondrejov.conf
```

with the following `metadata-ondrejov.conf` configuration file:

```
input {
   file {
      path => "/home/mazeltom/workers/active-learning/data/ondrejov-
metadata.csv"
      start_position => "beginning"
      sincedb_path => "/dev/null"
   }
}
filter {
```

```
   csv {
      separator => ","
      columns => ["id","dec","gratang","detector","expval","ob-
ject", "dichmir","chipid","ra","specfilt","label","date-obs"]
}
   mutate {convert => ["gratang","float"] }
   mutate {convert => ["expval","float"] }
   mutate {convert => ["specfilt","integer"] }
}
output {
   elasticsearch {
      hosts => "http://localhost:9200"
      index => "ondrejov"
   }
   stdout { codec => rubydebug }
}
```

3.4 Typical Workflow of an Active Learning Experiment

Typical active learning experiment proceeds in the following steps. The individual configuration files and their settings are described in detail in section 3.2.2.

- Prepare the data pool to be analyzed. You may consider creating a new folder at `https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse/filesystem/secured/index.xhtml?path=DATA%2F`. Just use the *New Folder* menu item at this website.

  Ideally, the spectra to be analyzed will be in a CSV file (e.g. `pool.csv`) prepared in a preprocessing step (preprocessing job) available at VOCLOUD (*Create job -> Restricted job -> Preprocessing*). You may also use a CSV file prepared in advance externally (e.g. the `pool.csv` file in the `AL-LAMOST-DR2` folder or the `ondrejov.csv` file in the `AL-ondrejov` directory at `https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse/filesystem/secured/index.xh-tml?path=DATA%2F`).

  Alternatively, a folder containing original FITS files may be used, as preprocessing capabilities are included in the active learning Python script. In preprocessing, a predefined wavelength region of raw FITS spectra is normalized and rebinned so that in the output `preprocessed.csv` file all spectra have values for the same wavelengths in the same wavelength region normalized to zero mean and unit variance. The CSV file is assumed to have a single-line header containing *'id'* in the first column and spectral wavelengths in all the others.

- Optionally, you may add a csv file containing the filenames of the spectra (e.g. the `poolnames.csv` file in the `active-learning` folder at `https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse/filesystem/secured/index.xhtml?path=DATA%2F`). This file is also expected to have a header *'id'*.

- Prepare a training set CSV file containing preprocessed spectra of already labelled samples. The file is expected to have a header containing *'id'* and *'label'* in the first two columns and spectral wavelengths in all the others. For the Be star project, the already labelled Ondřejov dataset is available (the `training_set.csv` file in the `active-learning` directory at `https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse/filesystem/secured/index.xhtml?path=DATA%2F`)).

  The labels are stored in numerical format, 1 corresponding to a single peak, 2 to a double peak, 0 represents "other" spectra.

- In case you would like to prepare your own training set, you may use any pool CSV file or original FITS files and label them in a separate labelling-only session (iteration 0). For LAMOST DR2 you may use the `pool.csv` file in `LAMOST-DR2` folder, for the Ondřejov dataset the `ondrejov.csv` file in the `AL-ondrejov` folder at `https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse/filesystem/secured/index.xhtml?path=DATA%2F`).

  The labelling session is started using the *Create job -> Standard jobs -> Active_learning* menu items in the VO-CLOUD menu. Precreated JSON configuration files for the labelling-only session are stored at `https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse/filesystem/secured/index.xhtml?path=CONFIG%2Factive_learning%2F`.

  For labelling data from a CSV file, you may use the `labeling_csv_lamost.json` configuration file, for labelling data from FITS files, you may use the `labeling_fits` JSON files.

  Start the job using the *Save and run* button at the bottom of the page. When it is finished, go to the *Details* and label the presented spectra using the radio buttons or the *"Alt + number"* keyboard shortcut. When you are done with the labelling, press the *Save* button to download/store the labels. The labels may then be uploaded to the folder of your choice using the *Manage filesystem -> Append new files -> Direct upload sequence*.

- Once you have your pool and training set data prepared, you are ready to start the actual active learning session (iteration 1). Again, use the *Create job -> Standard jobs -> Active_learning* sequence. Precreated JSON configuration files are available at VO-CLOUD. The `active_learning_`

`iteration1_simple.json` file contains the minimal necessary configuration, the `active_learning_iteration1_complex.json` file has many optional settings on top of that.

After setting the JSON configuration file, just hit the *Save and run* button to start the job. Select an appropriate session name (`"learning session name"` parameter in the JSON configuration). See section 3.2.2 for details.

- When the job is finished, go to the *Details* and label the presented samples (see section 3.1.3).

  Spectra from the `oracle.csv` file where the neural network detected the highest uncertainty (entropy) levels have white background, samples designated for performance estimation are highlighted in green, and potentially interesting (candidate) spectra are highlighted in yellow.

  Several functions have been added for users' comfort.

  Many times, when clicking or using the keyboard shortcuts too fast, you realize that you might want to go back and correct your label. For that purpose, the *Previous* button was included.

  In case you wish to see all the samples, i.e. their metadata, predictions and labels, just use the *All metadata* button.

  The *Save* button allows you to download the already assigned labels (*Download labels*), as well as the CSV file containing the processed spectra (*Download labelled spectra*). These files may then be uploaded to the folder of your choice in the VO-CLOUD filesystem and added in the next iteration to the Elasticsearch database (`"labels2database csv"` parameter in the JSON configuration file) and to the training set (`"training set addition csv"` parameter in the JSON configuration file).

  The *Performance* button allows you to view the performance estimate, i.e. the percentage of samples from the performance-estimate set where your label equalled the neural network prediction. The overall statistics, i.e. the number of samples in individual categories, is shown, as well.

  Several CSV files get created in the course of the iteration.

  These include the `candidate.csv` file containing the list of potentially interesting spectra, as well as their labels predicted by the network, the `oracle.csv`, `perf-est.csv`, and `to_label_csv.csv` files containing the IDs of the spectra presented for labelling, and the `spectra.txt`, `csv_spectra_file2.csv`, `raw_spectra_data.txt`, and `raw_spectra_wavelengths.txt` files containing the processed and raw spectral data.

Importantly, a file called `new_config.json` representing JSON configuration for the eventual next iteration is created, as well.

Finally, the `index.html` representing the displayed content and the comprehensive `results.zip` file are available in the job folder, as well.

- Further iterations are run similarly. The only difference is that at the beginning, labels from the previous iteration may be added to the training set (`training set addition csv` in the JSON configuration file) and to the Elasticsearch database (`labels2database csv` in the JSON configuration file). The iteration number is set by the `iteration num` parameter in the JSON configuration.
- Once the performance values get consistently acceptable, the session might be ended. To review the spectra of potentially interesting (candidate) objects, use the `"show_candidates":"yes"` setting in the JSON configuration.

Chapter 4

Discussion

4.1 User Experience

The user interface has been designed with the intention to be as user-friendly and flexible as possible. It was developed in close collaboration with its current users. Many features have been added or modified based on their wishes and comments.

4.1.1 User Interface Features

- In order to facilitate starting a new active learning job, the most commonly used JSON configurations have been prepared and uploaded to VO-CLOUD, so that users may load them according to the task they want to perform.

- Care was taken to make labelling as user-friendly and fast as possible. Users have two options, they may use mouse-click to select the radio button corresponding to their chosen label or they may use keyboard shotcuts (*Alt + number* corresponding to their choice).

- Care was taken to provide users with as much available information as possible. Therefore, not only the preprocessed spectra analyzed by the neural network, but also the raw spectra (the whole wavelength range) and metadata, such as object type (star, galaxy), spectral class, magnitude, right ascension, declination, etc.

- The system is very flexible. Users may choose what information should be displayed, number and names of label categories. On the other hand, almost fully automatic workflow may be used, as well.

- The spectra are displayed with a zoom-in, zoom-out option.

- Individual iterations are designed as separate jobs. Users may choose to abort or rerun them, save the labels in the Elasticsearch database, etc.
- Labels and labelled spectra may be downloaded as comma-separated-value files.
- Users have the option to display neural network performance estimate, as well as basic statistical analysis of its predictions.

4.2 Performance

The amount of data available for analysis keeps growing (see Table 4.1). Therefore, performance is one of the key parameters to keep in mind. As mentioned in [3], the use of GPU improves performance significantly. In our case, the convolutional network was trained on GPU GeForce GTX 980 with 2 048 CUDA cores and 4 GB of memory.

Table 4.1: LAMOST data release 6 version 1 [28]

Survey	**Objects**	**Stars**	**Galaxies**	**QSO**	**Unknown**
Pilot	958 679	832 886	8 197	1 386	116 210
First year	1 684 724	1 538 644	12 752	6 066	127 262
Second year	1 636 474	1 510 053	30 388	6 304	89 729
Third year	1 644 300	1 520 759	26 364	8 979	88 198
Fourth year	1 702 528	1 552 325	39 433	14 462	96 308
Fifth year	1 402 454	1 230 129	35 729	15 258	121 338
Sixth year	889 947	781 620	20 003	7 718	80 606
Total	**9 919 106**	**8 966 416**	**172 866**	**60 173**	**719 651**

The training of the neural network is the most time-consuming step taking approximately 8–15 minutes. In addition to that, the time necessary for retrieval of processed spectra from a CSV file, raw spectra import from original FITS files and retrieval of metadata from the Elasticsearch database was tested. The time for preparation of 100 and 1 000 randomly selected samples from LAMOST DR2 release for labelling and to run a complete active learning iteration including saving the predictions of the neural network, as well as the spectra of potentially interesting objects, in relation to the size of the pool is shown in Table 4.2.

Table 4.2: Time to prepare 100 and 1 000 spectra for labelling and to run a complete active learning iteration (mean ± SD in s, N = 10)

Data pool size	**100 spectra**	**1 000 spectra**	**Complete iteration**
1000	22.8 ± 1.0	172.3 ± 9.3	974 ± 166
10 000	24.9 ± 1.5	187.4 ± 11.5	785 ± 156
100 000	30.3 ± 0.9	230.7 ± 11.1	1015 ± 250
1 000 000	46.5 ± 3.0	324.6 ± 12.2	1264 ± 215
4 136 482	88.0 ± 3.4	545.9 ± 12.0	2046 ± 213

Further optimization of the code is planned to be done in near future.

4.3 Possible Future Improvements

In future, the system could/should be improved in many ways and additional capabilities could be added to it. For instance:

- A capability of saving and visualizing the state of the neural network in each iteration might be added.
- A separate VO-CLOUD job for just visualizing the spectra and their metadata, maybe also with the possibility of adding comments and labels, might be added.
- The Elasticsearch database could be exploited much more, e.g. it could be used for statistical analysis of the candidates, etc.
- The metadata table could be made interactive.
- Presently, CSV files are used to store most of the data. While this is very user-friendly, as data in this format may be easily visualized and analyzed, the data require a lot of storage space and more time to transfer. Alternatively, HDF5 file format with its efficient data compression, could be used. This option might be added.
- Performance of the system may be improved. Presently, complete active learning iterations involving the whole LAMOST database (millions of spectra) take more than 10 minutes. In course of the development, the system was continuously analyzed and many optimizations were applied in order to improve the performance. Nevertheless, many further improvements are still possible.
- User interface could be improved in order to make certain operations more user-friendly.

- Metadata visualization capabilities could be added to other available VO-CLOUD job types.
- An option to use other data sources and to create additional databases could be added.
- Docker technology was considered for easier deployment of the module. Nevertheless, due to the complex interconnections with the VO-CLOUD system, it would be necessary to containerize the whole VO-CLOUD system. This option might be added in the future.
- The infrastructure, in particular the possibility to visualize the preprocessed and raw spectra of promising candidates together with their metadata, may be used in combination with other VO-CLOUD capabilities, such as self-organizing maps [5], random decision forests [6], as well as dimensionality reduction approaches, such as the principal component analysis (PCA) [64] or t-distributed stochastic neighbour embedding (tSNE) [65].

The system is still under development. For current state, as well as future developments, please see
`https://vocloud-dev.asu.cas.cz/vocloud-betelgeuse` and
`https://github.com/tomasma/vocloud-active-learning`.

Conclusion

The main goals of the thesis have been fulfilled. An active learning module has been added to the current VO-CLOUD infrastructure allowing to conduct different scenarios of active learning of astronomical spectra in order to identify candidates with given spectral features and characteristic shapes. The engine allows interactive (re)classification/rejection of suggested candidates. It also allows to obtain all important metadata and to visualize different steps of data processing.

VO-CLOUD users will now have the possibility to apply active learning methods to interactively classify astronomical spectra and to visualize the results. The infrastructure, in particular the possibility to visualize the preprocessed and raw spectra of promising candidates together with their metadata, may be used in combination with other VO-CLOUD capabilities, such as self-organizing maps, random decision forests, as well as dimensionality reduction approaches, such as the principal component analysis (PCA) or t-distributed stochastic neighbour embedding (tSNE).

In future, many possible improvements might be envisaged. For instance, an option to visualize the state of the neural network could be added. The Elasticsearch database could be exploited much more, e.g. it could be used for statistical analysis of the candidates, etc.

Bibliography

[1] Koza, J. *Interactive Cloud-Based Platform for Parallelized Machine Learning of Astronomical Big Data*. Master's thesis, Czech Technical University in Prague, Faculty of Information Technology, Prague, 2017, doi: 10.5281/zenodo.573727. Available from: `https://zenodo.org/record/573727`

[2] Koza, J. *Design and Implementation of a Distributed Platform for Data Mining of Big Astronomical Spectra Archives*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, Prague, 2015, doi:10.5281/zenodo.44641. Available from: `https://zenodo.org/record/44641`

[3] Podsztavek, O. *Deep Learning in Large Astronomical Spectra Archives*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, Prague, 2015, doi:10.5281/zenodo.44641. Available from: `https://zenodo.org/record/44641`

[4] Mrkva, L. *VO-KOREL, Server for Astronomical Cloud Computing*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, Prague, 2012.

[5] Lopatovský, L. *Application of Self-Organizing Maps in Astroinformatics*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, Prague, 2014.

[6] Palička, A. *Application of Random Decision Forests in Astroinformatics*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, Prague, 2014.

[7] Newton, I. *Opticks: or, A Treatise of the Reflections, Refractions, Inflections and Colours of Light*. London: Royal Society, 1705.

[8] Fraunhofer, J. Bestimmung des Brechungs- und des Farben-Zerstreuungs — Vermögens verschiedener Glasarten, in Bezug auf die Vervollkommnung achromatischer Fernröhre. *Annalen der Physik*, volume 56(7), 1817: p. 282–287, doi:10.1002/andp.18170560706.

[9] Hearnshaw, J. *The Analysis of Starlight*. Cambridge: Cambridge University Press, 1986, ISBN 0-521-39916-5.

[10] Kirchhoff, G. R. *Gesammelte Abhandlungen*. Leipzig: Metzger & Wittig, 1882. Available from: `https://archive.org/details/gesammelteabhan01unkngoog/page/n12/mode/2up`

[11] Kirchhoff, G. R. Über das Verhältniss zwischen dem Emissionsvermögen und dem Absorptionsvermögen der Körper für Wärme and Licht. *Annalen der Physik und Chemie*, volume 109(2), 1860: p. 275–301. Available from: `doi:10.1002/andp.18601850205`

[12] Planck, M. Über das Gesetz der Energieverteilung im Normalspektrum. *Annalen der Physik*, volume 4(3), 1901: p. 553. Available from: `doi:10.1002/andp.19013090310`

[13] Siegel, R. *Thermal Radiation Heat Transfer, Fourth Edition*. CRC Press, 2001, ISBN 9781560328391.

[14] Bohr, N. On the Constitution of Atoms and Molecules, Part I. *Philosophical Magazine*, volume 26(151), 1913: p. 1–24. Available from: `doi:10.1080/14786441308634955`

[15] Tennyson, J. *Astronomical Spectroscopy: An Introduction to the Atomic and Molecular Physics of Astronomical Spectra (2nd Edition)*. WSPC, 2011, ISBN 9814291978.

[16] Morgan, W. W.; Keenan, P. C.; et al. *An Atlas of Stellar Spectra, with an Outline of Spectral Classification*. The University of Chicago Press, 1943.

[17] Morgan, W. W.; Keenan, P. C. Spectral Classification. *Astronomy and Astrophysics*, volume 11, 1973: pp. 29–50. Available from: `doi:10.1146/annurev.aa.11.090173.000333`

[18] Cannon, A. J.; Pickering, E. C. Spectra of Bright Southern Stars Photographed with the 13 Inch Boyden Telescope as Part of the Henry Draper Memorial. *Annals of Harvard College Observatory*, volume 28, 1901: p. 129.

[19] Hopkins, J. *Glossary of Astronomy and Astrophysics (2nd ed.)*. The University of Chicago Press, 1980, ISBN 978-0-226-35171-1.

[20] Morison, I. *Introduction to Astronomy and Cosmology*. Wiley, 2013, ISBN 978-1-118-68152-7.

[21] Porter, J. M.; Rivinius, T. Classical Be Stars. *Publications of the ASP*, volume 115, Oct. 2003: pp. 1153–1170. Available from: doi:10.1086/378307

[22] Balmer, J. Notiz über die Spectrallinien des Wasserstoffs. *Annalen der Physik und Chemie. 3rd series*, volume 25, 1885: p. 80–87.

[23] Ling, S.; Sanny, J.; et al. *OpenStax University Physics, University Physics Volume 3*. OpenStax, 2016, ISBN 978-1-947172-22-7. Available from: http://cnx.org/contents/af275420-6050-4707-995c-57b9cc13c358@12.2

[24] Struve, O. On the Origin of Bright Lines in Spectra of Stars of Class B. *Astrophysical Journal*, volume 73, 1931: p. 94. Available from: doi:10.1086/143298

[25] Korčáková, D.; Polster, J.; et al. Long-Term Spectroscopic Monitoring of B[e] Stars at the Ondřejov Observatory. *The B[e] Phenomenon: Forty Years of Studies, ASP Conference Series*, volume 508, 2017.

[26] AIASCR VO Services. Accessed: 2019-12-31. Available from: http://voarchive.asu.cas.cz/

[27] Astronomical Institute AS CR, Stellar Physics Department. Accessed: 2019-12-31. Available from: https://stelweb.asu.cas.cz/web/

[28] LAMOST Survey. Accessed: 2019-04-25. Available from: http://www.lamost.org/public/?locale=en

[29] Zhao, Y. Preparing First Light of LAMOST. 2009.

[30] McGlynn, J. M. A Primer on the FITS Data Format. Accessed: 2019-04-25. Available from: https://fits.gsfc.nasa.gov/fits_primer.html

[31] Ochsenbein, F.; Williams, R.; et al. IVOA Recommendation: VOTable Format Definition Version 1.3. *IETF*, 2011: p. 1. Available from: http://arxiv.org/abs/1110.0524

[32] Shafranovich, Y. Common Format and MIME Type for CSV Files. 2005. Available from: doi:10.17487/RFC4180

[33] Wells, D. C.; Greisen, E. W.; et al. FITS: A Flexible Image Transport System. *Astronomy and Astrophysics Supplement Series*, volume 44, 1981: p. 363–370.

[34] FITS Working Group (2016-07-22). Definition of the Flexible Image Transport System (FITS). Accessed: 2020-01-03. Available from: https://fits.gsfc.nasa.gov/fits_primer.html

[35] IVOA. VOTable Format Definition Version 1.3. Accessed: 2020-01-03. Available from: http://www.ivoa.net/documents/VOTable/20130920/REC-VOTable-1.3-20130920.html

[36] Heiter, U. Air-to-Vacuum Conversion. Accessed: 2019-04-25. Available from: http://www.astro.uu.se/valdwiki/Air-to-vacuum%20conversion

[37] Russell, S. J.; Norvig, P. *Artificial Intelligence: A Modern Approach*. Boston: Pearson Education, third edition, 2010, ISBN 978-0-13-604259-4.

[38] Mitchell, T. M. *Machine Learning*. New York City: McGraw-Hill, Mar. 1997, ISBN 0-07-042807-7, 432 pp.

[39] Hinton, J.; Sejnowski, T. *Unsupervised Learning: Foundations of Neural Computation*. MIT Press, 1999, ISBN 978-0262581684.

[40] Chapelle, O.; Schölkopf, B.; et al. *Semi-Supervised Learning*. Cambridge, Mass.: MIT Press, 2006, ISBN 978-0-262-03358-9.

[41] Settles, B. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[42] Culotta, A.; McCallum, A. Reducing Labeling Effort for Structured Prediction Tasks. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2005: p. 746–751.

[43] T. Scheffer, C. D.; Wrobel, S. Active Hidden Markov Models for Information Extraction. *Proceedings of the International Conference on Advances in Intelligent Data Analysis(CAIDA)*, 2001: p. 309–318.

[44] Dagan, I.; Engelson, S. Committee-Based Sampling for Training Probabilistic Classifiers. *Proceedings of the International Conference on Machine Learning (ICML)*, 1995: p. 150–157.

[45] H.S. Seung, M. O.; Sompolinsky, H. Query by Committee. *Proceedings of the ACMWorkshop on Computational Learning Theory*, 1992: p. 287–294.

[46] Hanisch, R.; Quinn, P. International Virtual Observatory Alliance [online]. *The IVOA*, accessed: 2020-02-06. Available from: http://ivoa.net/about/TheIVOA.pdf

[47] IVOA. What is the IVOA. Accessed: 2020-02-06. Available from: http://ivoa.net/about/what-is-ivoa.html

[48] Astropy Collaboration; Robitaille, T. P.; et al. Astropy: A community Python package for astronomy. *Astronomy and Astrophysics*, volume 558, Oct. 2013: A33, doi:10.1051/0004-6361/201322068, 1307.6212.

[49] Mckinney, W. pandas: a Foundational Python Library for Data Analysis and Statistics. *Python High Performance Science Computer*, 01 2011. Available from: https://www.dlr.de/sc/Portaldata/15/Resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf

[50] Chollet, F. Keras Documentation. Accessed: 2020-02-06. Available from: https://keras.io/

[51] Chollet, F. Introducing Keras 2. Accessed: 2020-02-06. Available from: https://blog.keras.io/introducing-keras-2.html

[52] Abadi, M.; Agarwal, A.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2015. Available from: http://download.tensorflow.org/paper/whitepaper2015.pdf

[53] Hunter, J.; et al. Matplotlib. Available from: https://matplotlib.org

[54] van der Walt, S.; Colbert, S. C.; et al. The NumPy array: a structure for efficient numerical computation. *CoRR*, volume abs/1102.1523, 2011. Available from: http://arxiv.org/abs/1102.1523

[55] Oliphant, T.; Peterson, P.; et al. SciPy. Accessed: 2020-02-06. Available from: https://scipy.org

[56] Cournapeau, D.; et al. SciPy. Accessed: 2020-02-06. Available from: https://scikit-learn.org/stable/

[57] Elasticsearch.py. Accessed: 2020-02-06. Available from: https://github.com/elastic/elasticsearch-py

[58] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, volume abs/1605.02688, May 2016: p. 19. Available from: http://arxiv.org/abs/1605.02688

[59] Bowyer, K. W.; Chawla, N. V.; et al. SMOTE: Synthetic Minority Oversampling Technique. *CoRR*, volume abs/1106.1813, 2011. Available from: http://arxiv.org/abs/1106.1813

[60] Nielsen, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015.

[61] Gitbook Machine Learning Questions. Accessed: 2020-02-06. Available from: https://ztlevi.github.io/Gitbook_Machine_Learning_Questions

[62] Elasticsearch history. Accessed: 2020-02-06. Available from: https://www.elastic.co/about/history-of-elasticsearch#group-1

[63] Elasticsearch as a No-SQL Database. Accessed: 2020-02-06. Available from: https://www.elastic.co/blog/found-elasticsearch-as-nosql

[64] Jolliffe, I.; Cadima, J. Principal component analysis: a review and recent developments. *Philos Trans A Math Phys Eng Sci*, volume 374(2065), 2016. Available from: doi:10.1098/rsta.2015.0202

[65] van der Maaten, L.; Hinton, G. E. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*, volume 9, 2008: pp. 2579–2605.

Appendix A

List of Abbreviations

ASU Astronomical Institute

CAS Czech Academy of Sciences

CNN Convolutional Neural Network

CSV Comma-Separated Values

EJB Enterprise Java Beans

FITS Flexible Image Transport System

GUI Graphical User Interface

Java EE Java Enterprise Edition

JSF Java Server Faces

JSON JavaScript Object Notation

LAMOST Large Sky Area Multi-Object Fiber Spectroscopic Telescope

PCA Principal Component Analysis

QSO Quasi-Stellar Object

RDF Random Decision Forest

REST Representational State Transfer

SOM Self-Organizing Map

SQL Structured Query Language

tSNE t-Distributed Stochastic Neighbor Embedding

UWS Universal Worker Service

VO Virtual Observatory

XHTML EXtensible HyperText Markup Language

XML EXtensible Markup Language

Appendix B

Contents of Enclosed CD

- `readme.txt` ..... brief description of CD content
- `src`
  - `impl` ..... source code directory
  - `thesis` ..... LaTeX source code of the thesis
- `text` ..... thesis text directory
  - `thesis.pdf` ..... thesis text in PDF format
  - `zzp.txt` ..... thesis task in plain text