



SCIENCE PLATFORM FOR MACHINE LEARNING OF BIG ASTRONOMICAL DATA – DATA ANALYSIS MODULES

Alisher Laiyk

Bachelor's thesis
Faculty of Information Technology
Czech Technical University in Prague
Department of Software Engineering
Study program: Informatics
Specialisation: Software engineering
Supervisor: RNDr. Petr. Škoda, CSc.
May 16, 2025



Assignment of bachelor's thesis

Title:	Science platform for machine learning of big astronomical data - data analysis modules
Student:	Alisher Laiyk
Supervisor:	RNDr. Petr Škoda, CSc.
Study program:	Informatics
Branch / specialization:	Software Engineering 2021
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2025/2026

Instructions

Science Platform is an emerging cloud-based technology for processing and exploratory analysis of big data sets in astronomy and earth sciences. The goal of the thesis is the design, implementation, testing and integration of several data analysis modules needed for performing selected machine learning methods on a large volume of astronomical spectra (namely active deep learning and dimensionality reduction as e.g. tSNE, PCA, UMAP) including various methods of preprocessing and data visualization. This thesis is complemented by the thesis of Olexandr Burakov focused on development of a cloud infrastructure allowing to launch these modules according to simple workflow manager through custom API.

The key tasks are:

- 1) Analyse data and parameters requirements of the typical machine learning procedures applied on spectra, mainly the active deep learning and tSNE.
- 2) Identify the best libraries for performing selected algorithms of pre-processing, active deep learning and dimensionality reduction as well as solutions for Big Data visual analysis.
- 3) Wrap such data analysis modules by the API defined in thesis of O. Burakov.
- 4) Prepare simple workflow scripts allowing to run the modules through the above mentioned API.
- 5) Demonstrate the correct functionality of workflows on a suitable data sets (e.g. LAMOST or SDSS spectra).



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

- 6) Integrate the modules with the help of O. Burakov into his cloud infrastructure
- 7) Discuss the performance and flexibility of your solution and suggest future improvements and extensions towards much larger and different type of astronomical data sets (e.g. light curves, images).

Recommended literature and suggested tools and libraries will be provided by supervisor.



Czech Technical University in Prague

Faculty of Information Technology

© 2025 Alisher Laiyk. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis: Laiyk Alisher. *Science platform for machine learning of big astronomical data – data analysis modules*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2025.

I would like to thank my supervisor RNDr. Petr Škoda CSc. for his help and advice in writing this thesis. My deep gratitude also goes to my family and friends, who supported me during my studies.

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 Section 2373(2) of Act No. 89/2012 Coll., the Civil Code, as amended, I hereby grant a non-exclusive authorization (licence) to utilize this thesis, including all computer programs that are part of it or attached to it and all documentation thereof (hereinafter collectively referred to as the "Work"), to any and all persons who wish to use the Work. Such persons are entitled to use the Work in any manner that does not diminish the value of the Work and for any purpose (including use for profit). This authorisation is unlimited in time, territory and quantity.

I declare that I have used AI tools during the preparation and writing of my thesis. I have verified the generated content. I confirm that I am aware that I am fully responsible for the content of the thesis.

In Prague on May 16, 2025

Abstract

The aim of this thesis was to implement analysis modules for applying selected machine learning methods on astronomical spectra. As a result of this thesis preprocessing, active deep learning, dimensionality reduction modules, and the front-end part for a cloud-based platform were implemented. All modules are runnable from the terminal. Moreover, preprocessing and active deep learning modules can be launched through a web interface. Using those modules enables to find interesting and unusual astronomical spectra.

Keywords web application, astronomical spectra, active learning, preprocessing, deep learning, dimensional reduction, React, Python, LAMOST DR2

Abstrakt

Cílem této práce bylo implementovat analytické moduly pro aplikaci vybraných metod strojového učení na astronomická spektra. Výsledkem této práce byla implementace modulů předzpracování, aktivního hlubokého učení, redukce dimenzí a front-endové části pro cloudovou platformu. Všechny moduly jsou spustitelné z terminálu. Moduly předzpracování a aktivního hlubokého učení lze navíc spouštět prostřednictvím webového rozhraní. Použití těchto modulů umožňuje vyhledávat zajímavá a neobvyklá astronomická spektra.

Klíčová slova webová aplikace, astronomická spektra, aktivní učení, předzpracování, hluboké učení, redukce dimenzí, React, Python, LAMOST DR2

Contents

Introduction	1
1 Analysis	2
1.1 Astronomical data	2
1.1.1 Astronomical spectroscopy	2
1.1.2 Astronomical spectrum	2
1.1.2.1 Electromagnetic spectrum	2
1.1.3 Storage format	3
1.1.3.1 FITS	3
1.1.3.2 HDF5	3
1.1.4 LAMOST data	3
1.2 Active deep learning	3
1.2.1 Deep learning	4
1.2.2 Convolutional neural networks	4
1.2.3 Class balancing	4
1.2.4 Active learning	4
1.2.4.1 Query Strategy	5
1.2.5 Performance estimation	5
1.3 Spectra preprocessing	5
1.4 Dimensionality reduction	6
1.5 VO-CLOUD	6
1.5.1 Active deep learning job	6
1.6 Requirements	7
1.6.1 Functional requirements	7
1.6.2 Non-functional requirements	8
2 Design	9
2.1 Analysis modules	9
2.1.1 Active deep learning module	9
2.1.1.1 Configuration file	10
2.1.2 Preprocessing module	11
2.1.2.1 Configuration file	11
2.1.3 dimensionality reduction module	11
2.1.3.1 Configuration file	11
2.2 Front-end	12
2.2.1 Angular	12

2.2.2	React	12
2.2.3	Next.js	13
2.2.4	Considerations	13
3	Implementation	14
3.1	Modules	14
3.1.1	Code modification	14
3.1.2	Active deep learning	14
3.1.2.1	Output	15
3.1.3	Preprocessing	16
3.1.3.1	Output	16
3.1.4	Dimensionality reduction	16
3.1.4.1	Output	16
3.1.5	Integration	17
3.2	Front-end	17
3.2.1	Styling	17
3.2.2	Communication with infrastructure	17
3.2.3	Pages	19
3.2.4	Job listing	19
3.2.4.1	Job creation	20
3.2.4.2	Job Detail	20
3.2.4.3	File system	23
3.3	Workflow	25
3.3.1	Launching	26
4	Discussion	28
4.1	Performance	28
4.2	Future improvements	28
5	Conclusion	30
A	Launching the programs	31
	Contents of the attachment	37

List of Figures

3.1	New platform model.	18
3.2	Home page and navigation bar.	19
3.3	Job listing.	19
3.4	Active learning job's creation form.	20
3.5	Job's information and files on job's detail page.	21
3.6	Spectrum labeling.	22
3.7	Plot of training data containing 60 spectra after applying t-SNE.	22
3.8	Performance estimation plot.	23
3.9	Plot of preprocessed spectrum.	23
3.10	Displaying file system.	25

List of Tables

List of Code listings

3.1	Example of application of t-SNE method.	16
3.2	Example of button	17
3.3	Example of plot rendering component.	24
3.4	Example of preprocessing job's configuration file.	25
3.5	Example of active learning job's configuration file for zero iteration.	26
3.6	Example of active learning job's configuration file for first and subsequent iterations.	26
3.7	Front-end Dockerfile.	27
3.8	Front-end Docker Compose file.	27

List of Code listings	x
-----------------------	---

A.1 Example of module launching.	31
--	----

List of abbreviations

API	Application Programming Interface
DOM	Document Object Model
CNN	Convolutional neural network
FITS	Flexible Image Transport System
GPU	Graphics processing unit
HDF5	Hierarchical Data Format Version 5
HTML	HyperText Markup Language
JSON	JavaScript Object Notatio
LAMOST DR2	The Large Sky Area Multi-Object Fiber Spectroscopic Telescope Data Release 2
REST API	Representational State Transfer Application Programming Interface
t-SNE	t-Distributed Stochastic Neighbor Emedding
XHTML	EXtensible HyperText Markup Language

Introduction

The datasets in astronomy are vast, with some observatories gathering terabytes of information daily. This sheer volume exceeds the capabilities of manual processing and analysis. Therefore, astronomers use machine learning algorithms to identify valuable data. [1]

The VO-CLOUD platform provided opportunities for processing and analyzing large astronomical data [2, 3]. Currently, it requires refactoring.

The aim of this thesis was to design, implement, and integrate several data analysis modules necessary to apply selected machine learning methods on a large dataset of astronomical spectra. That is, pre-processing, active deep learning, and dimensionality reduction. In addition, a new front-end was developed. This thesis is complemented by the thesis of Olexandr Burakov, who developed the platform's infrastructure. The preprocessing and active deep learning modules were integrated into the back-end, while the dimensionality reduction module was left standalone.

This thesis commences with an analytical chapter that provides an overview of astronomical data, a review of module analyses, and a discussion of the VO-CLOUD platform. The second chapter details the configuration of each module and the selection of technologies used in the front-end. The third chapter discusses the implementation and integration of analysis modules, alongside the development of the front end. Finally, a review of the implementation, its performance, and potential improvements in features are given.

In this thesis, the code [4] developed by Ing. Ondřej Podsztavek during his bachelor's thesis [5] and writing the article [6] was utilized.

Chapter 1

Analysis

1.1 Astronomical data

In this section, the astronomical spectrum and its properties are discussed, and file formats are reviewed to store such data.

1.1.1 Astronomical spectroscopy

Spectroscopy is one of the most important tools in astronomy for exploring the universe. It enables the identification of chemical compositions and physical properties of astronomical sources. [7]

1.1.2 Astronomical spectrum

Spectrum is a plot that shows the intensity of emitted light over a range of energies. Each spectrum contains a wide variety of data. [8]

1.1.2.1 Electromagnetic spectrum

White light(visible or optical light) can be divided into its constituent colors, resulting in the rainbow. A rainbow is a continuous spectrum that displays the varying light energies present in white light. The electromagnetic spectrum includes a variety of light energies beyond white light. It covers all energies of light, extending from low-energy radio waves, through microwaves, infrared, optical light, ultraviolet, high-energy X-rays and gamma rays. [8]

1.1.3 Storage format

The next subsections describe the files format in which astronomical data are often stored.

1.1.3.1 FITS

Flexible Image Transport System (FITS) is the data format most commonly used in astronomy for transporting, analyzing and storing scientific data files. FITS is mainly used to store scientific data set presented as multidimensional arrays and 2-dimensional tables structured into rows and columns containing data. [9]

1.1.3.2 HDF5

The Hierarchical Data Format version 5 (HDF5) is a file format supporting massive, complex, and diverse data. HDF5 can be seen as a file system contained within a single file. In HDF5, directories are called **groups** and files are called **datasets**. Groups may include other groups or datasets inside them. Datasets include the actual data in the file, and are often contained in other groups. Moreover, each group and dataset may have metadata, which describe the data within them. [10]

1.1.4 LAMOST data

The Large Sky Area Multi-Object Fiber Spectroscopic Telescope (LAMOST) is located in China. The second LAMOST data release 2 (DR2) contains 4,136,482 spectra. The wavelengths of the LAMOST spectra cover the range of 3100-9100 angstrom(\AA), where one angstrom is equal to 10^{-10} meters. The LAMOST DR2 contains:

- 3,784,461 stars.
- 37,206 galaxies.
- 8,630 quasars.
- 306,185 unknown objects. [11]

1.2 Active deep learning

Using standard machine learning algorithms on astronomical spectra data without a proper training set would give an inaccurate result. This is why the convolutional neural network (CNN) should be applied, as it has been used effectively in astronomy and astrophysics. However, applying CNN alone is insufficient, they must be combined with class balancing and active learning for effectiveness. [6]

1.2.1 Deep learning

Traditional machine learning approaches had constraints in processing raw data. For years, developing an effective machine learning system required thorough research to create a transformation from raw data to a representation or feature vector, which is utilized by the learning system for data classification. To remove this constraint, representation learning is used; it can take raw data and automatically extract representations. Deep learning approaches are a form of representative learning with multi-layer representations, consisting of non-linear modules, where each module transforms the representation from one level to a higher-level representation. [12]

Numerous deep learning applications use feedforward neural networks, which aim to map inputs and outputs of consistent size. To move between layers, a group of units computes a weighted sum of their inputs derived from the preceding layer and then applies a linear function to this result. [12]

1.2.2 Convolutional neural networks

The convolutional neural network (CNN, ConvNet) is a type of deep feedforward network that handles data in multiple arrays. Typical CNN is organized as a series of stages. The first few stages consist of two types of layers: convolutional layers and pooling layers. In a convolutional layer, units are formed in feature maps, where each unit connects to specific areas in the feature maps of the earlier layer through a collection of weights, which are named a filter bank. The output of the locally weighted sum is fed into a non-linear function. One instance of a filter bank is shared between all units in one feature map. Each feature map in a layer uses distinct filter banks. The feature map carries out a filtering operation as a discrete convolution. In the pooling layer, similar features are combined into one. [12]

1.2.3 Class balancing

Usually, labeled target spectra are less common compared to labeled non-target spectra. Hence, the labeled training data will be unbalanced and the target spectra will represent a small minority. To overcome this problem, the synthetic minority oversampling technique (SMOTE) will be applied. SMOTE will equate the count of labeled target spectra with the count of labeled non-target spectra. [6]

1.2.4 Active learning

Active learning is a specialized subfield of machine learning in which the learner selects the most informative sample for labeling. This concept not only improves algorithm performance, but at the same time reduces the amount of

training samples, which is essential detail since sometimes obtaining large amount of training samples is challenging. Active learning solves this problem by querying the oracle to label selected unlabeled samples. [13]

There are several types of active learning scenarios, in this thesis work **pool-based active learning** is used, where the learner selects an unlabeled sample from a static pool [13].

1.2.4.1 Query Strategy

Active learning requires the evaluation of the informativeness of the unlabeled sample. The most basic and widely used query strategy is the *uncertainty sampling*. In this strategy, the learner queries the samples where it is least confident. The uncertainty measure will be calculated through the entropy:

$$x_{\text{ENT}}^* = \arg \max_x - \sum_i P(y_i | x; \theta) \log P(y_i | x; \theta).$$

where y_i covers all possible labels. [13]

1.2.5 Performance estimation

Monitoring the CNN's performance is necessary for determining when to end active learning iterations. Active learning stops when newly labeled spectra no longer increase CNN performance. Performance precision estimated as:

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

where TP (truly positive) represents the count of correctly predicted target spectra and FP (false positive) represents the count of incorrectly predicted target spectra. [6]

1.3 Spectra preprocessing

To utilize active deep learning, spectral data have to be structured as a matrix, where each row corresponds to an individual spectra sample, while each column corresponds to a flux measurement at specific wavelengths. Thus, each spectrum needs to be transformed to have the same wavelengths [5]. A specific interval of wavelengths and uniform points within it will be selected.

After that, the min-max normalization is applied, which will scale the spectral flux into a unit-less range from -1 to 1 using equation [6]:

$$x' = 2 \cdot \frac{x - \min(x)}{\max(x) - \min(x)} - 1$$

Where x is an unscaled spectrum and x' is a scaled spectrum. This was done for the following reasons:

- Spectra are classified according to their forms [6].
- The value of $[-1, 1]$ is suitable for applying CNN [6].
- Spectra will map to the same space after dimensionality reduction [5].

1.4 Dimensionality reduction

Following preprocessing, each spectrum is represented as a point in high-dimensional space. In order to better understand the data, dimensionality reduction should be used. [5]

t-Distributed Stochastic Neighbor Embedding(t-SNE) dimensionality reduction algorithm that transforms each point from a high-dimensional space to a point in 2-dimensional or 3-dimensional space. t-SNE is like Stochastic Neighbor Embedding(SNE), but it applies a heavy-tailed distribution in a low-dimensional space, which solves the disadvantages of SNE. [14]

1.5 VO-CLOUD

VO-CLOUD is a distributed system for applying machine learning algorithms on big astronomical data. The user invokes these algorithms as computational tasks called jobs [2, 3]. Currently, VO-CLOUD requires refactoring because certain functionalities are non-operational(such as using Jupyter [15], launching jobs), and adding new functionalities to the system is impossible. The platform's infrastructure was developed by Olexandr Burakov.

The front-end part needs to be refactored for the following reasons:

- VO-CLOUD uses XHTML that is no longer supported [16].
- The back-end is responsible for generating HTML for the front-end, which means that any change to the user interface requires modifications in the back-end code.
- Jobs at the end of their completion must produce HTML code that displays the result. [2]

1.5.1 Active deep learning job

One of the VO-CLOUD's jobs was an active deep learning job developed by MUDr. Tomáš Mazel Ph.D. This job used a CNN model [4] developed by Ing.Ondřej Podsztavek. For saving the result CSV [17] files were used, while the metadata was stored using Elasticsearch [18]. [19]

However, this job also requires refactoring primarily because the code [20] is difficult to maintain:

- Functions contain too much code and accept too many parameters (sometimes unused).
- The code is highly nested.
- Code is not logically divided to smaller function or modules.

In addition, storing large astronomical datasets in a CSV file can result in excessive memory usage and considerably slow down the processes of reading and writing [21].

1.6 Requirements

Requirements are often divided to two types:

1. Functional: outlines the types of functionalities the system must possess.
2. Non-functional: outline the necessary quality constraints that functionalities must satisfy. [22]

1.6.1 Functional requirements

1. The user can launch modules from the terminal.
2. The user can create preprocessing and active deep learning tasks (jobs) using JSON configuration files and launch them through the web interface.
3. The user can abort running jobs through the web interface.
4. The user can delete jobs through the web interface.
5. The user can view a list of all jobs.
6. The user can download the results of successfully finished jobs.
7. The user can upload, download, and delete files and directories from the platform's storage.
8. After completing the active deep learning job, the user can see the plot of the raw and preprocessed spectrum and some of their metadata.
9. After completing the active deep learning job, the user can label samples and evaluate the job's performance.

1.6.2 Non-functional requirements

1. The preprocessing and active deep learning modules are integrated into the platform infrastructure.
2. If the displayed job is still running, the system will check every 10 seconds the status of the job until the job is finished.
3. The user can not delete the processing job.

Chapter 2

Design

In this chapter, module analysis and front-end design will be discussed.

2.1 Analysis modules

Just like in VO-CLOUD [2, 3] each module is configured with a JSON file and outputs some files. The preprocessing and active deep learning module can be launched through the web interface. All modules can be launched from the terminal. Nevertheless, it is recommended to use the web interface.

In the preprocessing and active deep learning module, the code [4] developed by Ing. Ondřej Podsztavek during his bachelor's thesis [5] and writing the article [6] will be used.

2.1.1 Active deep learning module

In this module CNN model [4] developed by Ing. Ondřej Podsztavek will be used. It has an input layer containing 140 units and an output layer containing 3 units with softmax activation. In the convolutional layer all filters with size 3 pixels and without padding. The initial two layers contain 64 filters, the subsequent two contain 128 filters, and the final pair of layers features 256 filters. Following every two convolutional layers, there is a max-pooling layer with a size of two pixels with a stride of two. After the final max-pooling layer, there are two fully-connected layers, each containing 512 units, with a dropout rate of 0.5 applied during training. Model is implemented using **Tensorflow** framework. [5]

The TensorFlow framework is developed by Google. It provides a wide range of tools and libraries for machine learning. TensorFlow has been applied both within computer science and across other scientific fields. [23]

2.1.1.1 Configuration file

Some of the following parameters were also present in the previous implementation of the active deep learning module [19]. Parameters for model training, prediction, and saving, and a path to the performance estimation file were added.

- **iteration:** iteration number.
- **training_data_path:** path to the HDF5 file that contains training data.
- **pool_data_path:** path to the HDF5 file that contains the pool data.
- **classes:** list used for spectra classification.
- **candidate_classes:** list of candidate classes.
- **training_data_to_add_path:** path to HDF5 file containing the oracle spectra.
- **oracle_data_to_add_path:** path to JSON file containing labels for oracle spectra. Those spectra and labels will be added to the training data.
- **oracle_batch_size:** the number of spectra that need to be labeled by the oracle. The default value is 100.
- **perf_est_batch_size:** the number of spectra to calculate the performance estimation. The default value is 10.
- **perf_est_list_path:** path to JSON file consisting of performances from previous iteration. Is required starting from second iteration.
- **show_candidates:** if true, shows spectra that were predicted as candidate class, otherwise does not. The default value is false.
- **save_model:** if true, saves model. The default value is false.
- **epochs_train:** number of iterations to train the model, one interaction goes through the entire training data [24]. The default value is 6400.
- **batch_size_train:** sample count for every gradient update [24]. The default value is 64.
- **min_delta_train:** change in the monitor metrics that can be treated as improvement [25]. The default value is 0,004.
- **patience_train:** number of epochs without improvements after which model training is terminated [25]. The default value is 10.

- **batch_size_predict:** samples count per batch [24]. The default value is 16384.

Similar to the previous implementation of the active deep learning module [19], iterations start from 0. The 0 iteration is distinct, serving specifically to prepare the initial training data without applying deep learning. The default values for model training and prediction were hard-coded in code [4], which is why they were chosen as default ones.

Provided HDF5 files must have **filenames**, **wave**, **fluxes** datasets. Additionally, for training data **labels** dataset is required, which contains integer corresponding to the **classes** indexes.

2.1.2 Preprocessing module

For spectra transformation, the **NumPy** 1.25.0 library is used. It is a scientific computing library in Python that provides a multidimensional array and various fast operations on them. [26, 27]

For spectra scaling, the **scikit-learn** 1.6.1 library is used, which provides various machine learning algorithms. [28, 29]

2.1.2.1 Configuration file

Preprocessing module configuration parameters:

- **wave_start_point:** starting wavelength in angstroms.
- **wave_end_point:** ending wavelength in angstroms.
- **wave_point_count:** count of fluxes within the selected wavelength interval.
- **data_dir_path:** path to directory that contains LAMOST DR2 spectra in FITS files.

2.1.3 dimensionality reduction module

For dimensionality reduction **scikit-learn** 1.6.1 [29] library is used.

2.1.3.1 Configuration file

The configuration file will contain only two parameters:

- **data_path:** path to HDF5 containing pre-processed spectra.
- **classes:** classification classes.

Provided HDF5 files must have **fluxes** and **labels** datasets.

2.2 Front-end

In the next subsections, various web frameworks and technologies are discussed.

The design of the web interface will be similar to the design of the VO-CLOUD [2, 3, 19] platform, which will help current users transition to the new web interface without any need to relearn the user interface.

2.2.1 Angular

Angular is a web framework created by Google that relies on TypeScript. It employs two-way data binding, allowing real-time synchronization between the data model and the user interface. Through the use of dependency injection, developers can craft modular components that are both easy to maintain and test. Additionally, Angular provides built-in libraries that offer extensive features like routing, form management, and more. [30, 31]

Angular's learning curve is steep due to its complex architecture. Angular applications generally have larger bundle sizes of data compared to other frameworks due to the integration of numerous libraries and features. This results in slower loading times. To solve these issues, specific strategies or patterns must be applied. Using Angular for a small application may lead to poor performance. Angular is frequently updated, leading to refactoring of existing code. [31]

2.2.2 React

React is a library developed by Meta company for building user interfaces. It uses JavaScript Syntax Extension (JSX), which combines JavaScript and HTML, and is used for creating React components. These components are reusable and can be combined with other components to form a complex user interface. Data between components is passed in one direction, from the parent component to the child component. One of the important React features is a virtual DOM(Document Object Model), which is an instance of the original DOM [32]. Updating the entire DOM is slow, but the virtual DOM only updates the specific parts, making updates much more efficient. [33, 34].

React is regularly updated, requiring developers to always monitor changes. Due to frequent updates, the React documentation may not be immediately updated. Without proper optimization, components may cause unnecessary re-renders. React serves as a library for building user interfaces, not a full-fledged framework, which leads to the necessary integration of certain technologies and libraries. [34, 35]

2.2.3 Next.js

Next.js is a framework developed by Vercel for building web applications using React components. In a Next.js application, the HTML page is rendered on the server side, and then React makes it interactive on the client side using JSON data and JavaScript. In addition, it supports statically generated web pages. Instead of manually creating routes for the web application, they are generated from the folder structure of the web application. When a page is loaded, not all scripts are downloaded, they will be downloaded as needed, which improves application performance. Next.js enables the creation of API Routes, which can be used to create your own endpoints. [34, 36]

Similar to Angular and React, Next.js receives regular updates, requiring developers to spend more time learning new features and refactoring existing projects. Next.js has a hard learning curve, it introduces some concepts that are hard to learn. Connecting routing logic directly to pages complicates page maintenance. As the application expands, the build time increases. [34, 37]

2.2.4 Considerations

Currently, the front-end needs to support only a certain number of functionalities:

1. Displaying jobs with pagination.
2. Using the platform's file system.
3. Creating, launching, aborting, and deleting a job.
4. Downloading the result of the job.
5. Spectra labeling.

These functionalities do not require complex logic. Thus, using Next.js or Angular framework for their development is not necessary. Hence, React was chosen to implement the front-end.

Moreover, **Vite** build tool will be used, which provides faster development. It has a dev server containing superior features compared to native JavaScript modules [38]. The **Tailwind CSS** framework is applied for styling, offering classes that can be combined to create new designs [39]. JavaScript will be replaced with **TypeScript** [40] due to its static typing.

Implementation

In this chapter, data analysis modules and the front-end implementation will be discussed.

3.1 Modules

Each module takes two parameters: the path to the configuration file and the path to the directory where the result of the module will be saved.

3.1.1 Code modification

The code written by Ing. Ondřej Podsztavek requires modification, since it is programmed to process the wavelength of the spectra within the interval from 6519Å to 6732Å, using 140 uniform points along this range, and classifies the spectra into three distinct classes.

3.1.2 Active deep learning

CNN model [4] was updated, now it works with different numbers of classes, waves, and uniform points. The model training and prediction parameters are now configurable. After loading the configuration, the workflow of the module for the first and further iterations:

1. Reads training and pool data. Duplicate data is removed. Additionally, any training data present within the pool dataset is also excluded from the pool. The consistency between waves from training and pool data is verified to ensure they match.
2. After successful loading, training data is balanced, then the model training and prediction starts.

3. For each spectrum, the entropy is calculated, and label with the highest probability is taken.
4. Selecting spectra with the highest entropy for oracle labeling. Identifying spectra predicted as candidate classes and a random subset of those spectra for performance estimation.
5. Necessary files are created, and the result is saved to an HDF5 file.

Workflow for zero iteration is simpler; it only takes the first count of the spectra from the pool data for labeling and creates files.

3.1.2.1 Output

After successful completion, several files were created:

1. **training_data.h5**: HDF5 file containing training data from current job.
2. **new_config.json**: configuration file for next iteration.
3. **dim_reduc.json**: contains data for scatter plot after t-SNE applying on training data. Will not be created in zero iteration.
4. **perf_est_list.json**: contains performances from previous iterations. Will not be created in zero iteration.
5. **result.h5**: HDF5 file containing the result of the job with the following datasets:
 - **filenames**: contains the filenames of LAMOST DR2 spectra.
 - **wave**: contains preprocessed waves.
 - **fluxes**: contains preprocessed fluxes.
 - **labels**: contains labels predicted by CNN.
 - **entropies**: contains entropies(informativeness) of spectra.
 - **oracle_indexes**: contains the indexes of spectra, which were selected for labeling.
 - **perf_est_indexes**: contains the indexes of spectra, which were selected to calculate the performance estimation.
 - **candidate_indexes**: contains the indexes of spectra, which were predicted as candidate class. Will not be created if the `show_candidate` parameter in the configuration is false.

Using indexes from `oracle_indexes`, `perf_est_indexes`, `candidate_indexes` on the `filenames` or `fluxes` dataset, you will obtain the corresponding spectrum filename and fluxes.

3.1.3 Preprocessing

Spectrum wave and fluxes are now configurable. After loading the configuration, the workflows of module:

1. Transform new wave for spectra.
2. Reads each spectrum filename, wave and fluxes from FITS file in the provided directory, and transform fluxes.
3. After reading and transforming all spectra, applies fluxes scaling.
4. Saves the result.

3.1.3.1 Output

The result will be saved to HDF5 file **result.h5** with the following datasets:

1. **filenames:** contains the names of spectra.
2. **wave:** contains preprocessed waves.
3. **fluxes:** contains a list of preprocessed fluxes.

3.1.4 Dimensionality reduction

Workflow of the module after loading the configuration:

1. Reading fluxes and labels from the provided HDF5 file.
2. Applying t-SNE algorithm on fluxes.
3. Constructing the plot.
4. Saving the result.

```
from sklearn.manifold import TSNE

tsne = TSNE(random_state=42)
fluxes_embedded = tsne.fit_transform(fluxes)
```

■ **Code listing 3.1** Example of application of t-SNE method.

3.1.4.1 Output

The result of the module is a picture of a constructed scatter plot after applying t-SNE and **dim_reduc.json** file containing data for recreating the scatter plot.

3.1.5 Integration

After the modules were implemented, with the help of Olexandr Burakov they were integrated into an infrastructure he had developed. The dimensionality reduction module was not directly integrated, but is invoked inside the active deep learning module to visualize the training data. Together with Olexandr Burakov we refactored the module's code to match his coding style, added configuration validation and documentation.

LAMOST DR2 FITS files are divided into many directories. In the infrastructure, all these directories are located in one root directory. When writing the configuration for the preprocessing job, the path to the directory containing LAMOST DR2 FITS file needs to be written relative to the root directory, for example `/name_of_directory`.

Moreover, spectra that were selected for oracle labeling, performance estimation, and were predicted as candidate classes, are saved to the database using the API that was defined by Olexandr Burakov.

3.2 Front-end

In this section, the structure and functionalities of the implemented front-end are discussed. Moreover, active deep learning job has been shortened to active learning job.

3.2.1 Styling

As was said before, for styling Tailwind CSS [39] framework is used. The following code demonstrates styling a button using Tailwind classes:

```
<button className="p-2 bg-blue-600 text-white rounded
↪ cursor-pointer disabled:opacity-50 hover:bg-blue-800">
  button text
</button>
```

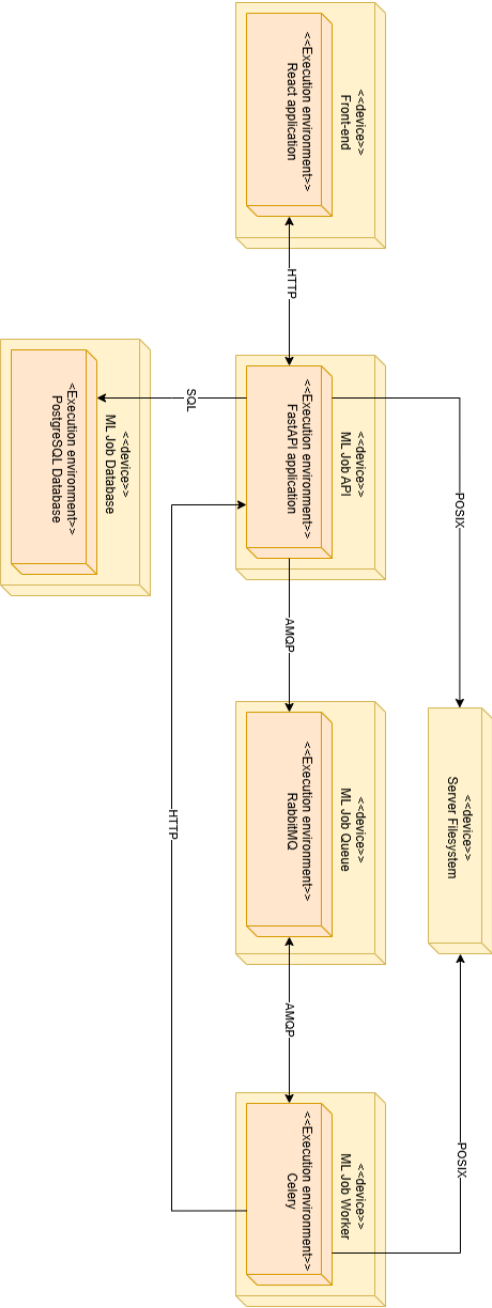
■ Code listing 3.2 Example of button

The button that is produced has a blue background with white text and rounded edges. When hovered over, the background becomes darker and the cursor changes to a pointer. If the button is disabled, its opacity decreases to 50 %.

3.2.2 Communication with infrastructure

All communication between the front-end and the platform infrastructure occurs over a RESTful API [41]. When the front-end sends an HTTP [42] re-

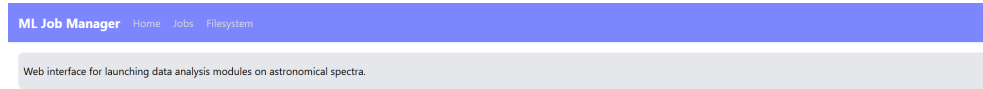
quest(creating a new job, starting a job, uploading a file, etc), the infrastructure accepts the request and executes the corresponding workflow. Figure 3.1 demonstrates the entire platform model.



■ **Figure 3.1** New platform model.

3.2.3 Pages

The navigation bar is fixed on all pages to navigate through the web interface.



■ **Figure 3.2** Home page and navigation bar.

3.2.4 Job listing

The list of jobs is displayed in a table using pagination and shows the phase, type, identifier, label, timestamps, and execution duration of each job. For enhanced clarity, jobs are color-coded according to their phase:

- **PENDING** – white. The job is created, but is not started.
- **PROCESSING** – yellow. The job is running.
- **ABORTED** – purple. The job was aborted during the PROCESSING phase.
- **ERROR** – red. The job ended with an error.
- **COMPLETED** – green. The job successfully completed.

If at least one job in the list is in the PROCESSING phase, every 10 seconds, the job's phase will be checked until the job is finished. To view more detail about job's detail,

Phase	Type	Job id	Label	Created at	Started at	Ended at	Execution duration	View
PROCESSING	ACTIVE_ML	2f794441-a505-42c2-8a79-9bd384f8412a	active_learning_3	10.05.2025, 14:24:28				View
PENDING	DATA_PREPROCESSING	40d779eb-f065-4810-ab21-4015a808f030	preprocessing_2	10.05.2025, 14:18:00				View
ABORTED	ACTIVE_ML	aa5b40f1-33b4-41ca-8b2f-ce4992d2fd49	active_learning_2	10.05.2025, 14:16:07				View
ERROR	DATA_PREPROCESSING	29ce9de9-29bf-4dc6-95d1-5e72e4e036d4	preprocessing_1	10.05.2025, 14:08:22	10.05.2025, 14:08:23	10.05.2025, 14:08:23	0 sec	View
COMPLETED	ACTIVE_ML	dfe273c6-aab1-4b69-b186-ea2b7430cb10	active_learning_1	10.05.2025, 14:07:52	10.05.2025, 14:07:59	10.05.2025, 14:08:04	5 sec	View

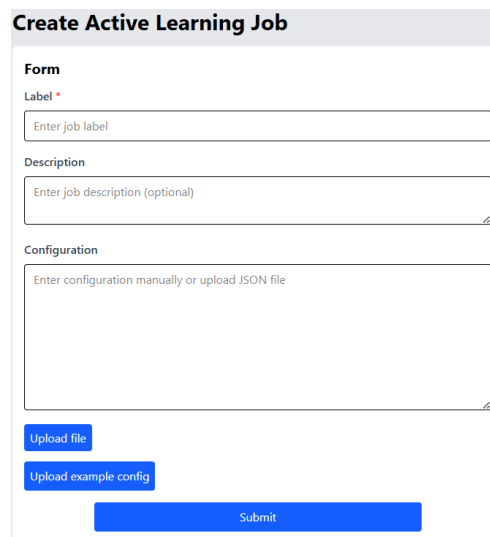
■ **Figure 3.3** Job listing.

3.2.4.1 Job creation

To create a new job, click on the appropriate button above the job table. After that, the form creation needs to be filled, label and configuration fields are required, description is optional. The user can fill configuration in one of the following ways:

- Write the configuration directly.
- Upload the configuration file. After uploading, the contents of the file are displayed for the user to review and edit.
- Upload the template and then edit it. The template will contain all configuration fields for this job.

An error message will be displayed if the label is empty, or the configuration is empty or invalid. After successful job creation, the user will be redirected to the job's detail page.



Create Active Learning Job

Form

Label *

Enter job label

Description

Enter job description (optional)

Configuration

Enter configuration manually or upload JSON file

Upload file

Upload example config

Submit

■ **Figure 3.4** Active learning job's creation form.

3.2.4.2 Job Detail

The user can view the job details on this page and control its execution: starting, aborting, and deleting. Deletion is not allowed if the job is in the PROCESSING phase. After the job is finished log file is created, and regardless of whether it is in ABORTED, ERROR, or COMPLETED phase, the job can not be restarted. Similar to the job listing page, if the job is in the PROCESSING phase, every 10 seconds, the job's phase will be checked until the job is finished. Moreover, the user can view and download files in the job directory.

Once a job has been successfully finished, it transitions into the COMPLETED phase. In this phase, the user can view the job results. For the preprocessing job, the user can only download the result file. However, the active learning job detail page has additional functionalities.

ML Job Manager Home Jobs Filesystem

job_label

Job id	Type	Phase	Created at	Started at	Ended at	Execution duration
1c4c42a0-914f-4b3a-915a-998aaeed38d	DATA_PREPROCESSING	COMPLETED	15.05.2025, 22:04:16	15.05.2025, 22:04:20	15.05.2025, 22:04:24	3.16 sec

Description:
You're reading the job's description

[RUN](#) [ABORT](#) [DELETE](#)

Job's directory:
/JOBS/job_job_label_1c4c42a0-914f-4b3a-915a-998aaeed38d

Job's files:

Filename	Size	Download
result.h5	3674096 B	Download
config.json	120 B	Download
log.txt	31 B	Download

■ **Figure 3.5** Job's information and files on job's detail page.

In the COMPLETED phase of the active learning job, the user reviews several spectra and can label them. Each spectrum will be colored depending on its set:

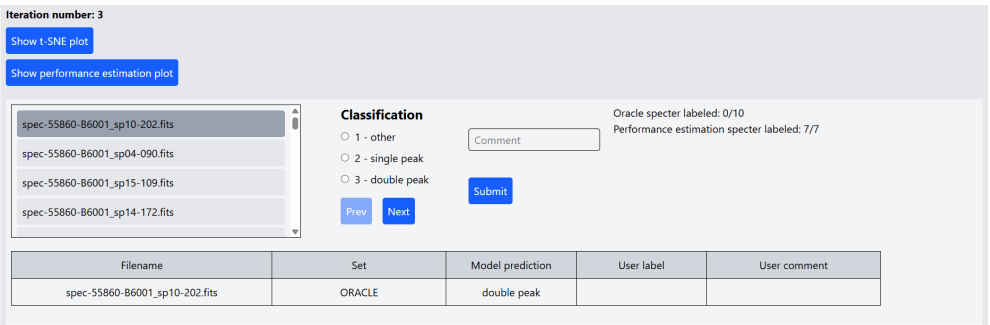
- **ORACLE** – gray. Spectra selected for oracle labeling.
- **PERFORMANCE_ESTIMATION** – orange. Randomly chosen spectra to evaluate performance.
- **CANDIDATE** – blue. Spectra predicted as candidate classes.

The user labels the spectra using radio buttons and has the option to add a brief comment to each spectrum. For navigating between spectra *prev* and *next* buttons are used. Additionally, upon selecting the radio button, the next spectrum in the list will be automatically selected. After clicking *submit* button following steps is happens:

- Saving labeled and commented spectra to the database.
- If any oracle spectrum were labeled, names and labels of all user labeled oracle spectra will be saved to the **oracle_data.json** file, which is used in the next iteration to adding oracle spectra to the training data.
- If all spectra from PERFORMANCE_ESTIMATION set were labeled, performance for current iteration will be calculated and the result will be saved to **perf_est_list.json** file.

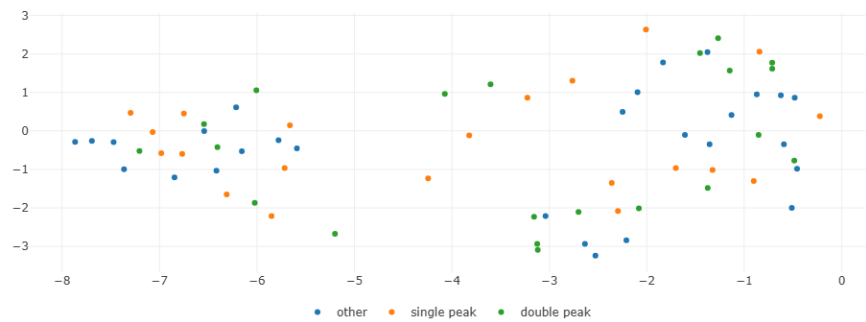
It is not mandatory to label every oracle spectrum, at least one is enough. However, for estimating performance, all spectra from PERFORMANCE_ESTIMATION set need to be labeled. If the user attempts to start the job for the next iteration without estimating the performance for the current iteration, the job will fail.

Information about labeling displays how many spectra from ORACLE and PERFORMANCE_ESTIMATION set are labeled and saved. The user can view a scatter plot of training data after t-SNE is applied, and view a plot of performance by clicking corresponding buttons, those buttons are not presented if job's iteration is 0.

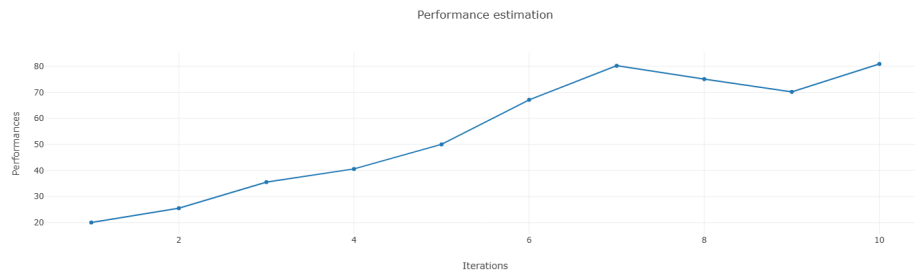


■ **Figure 3.6** Spectrum labeling.

Labels for figure 3.7 and values for figure 3.8 were randomly generated for illustration purposes.

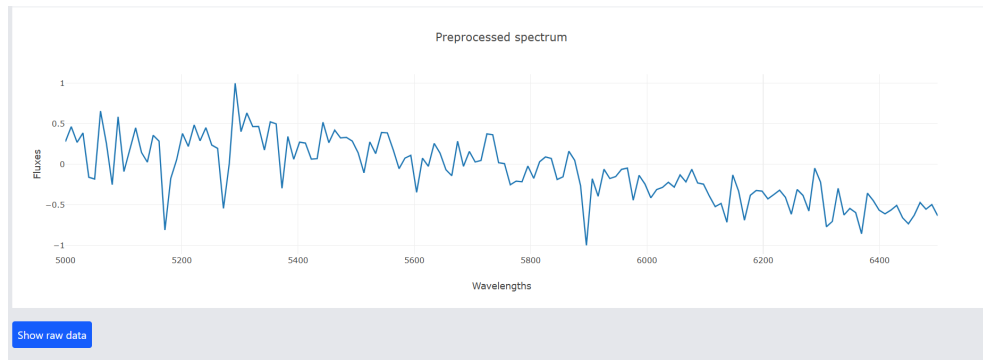


■ **Figure 3.7** Plot of training data containing 60 spectra after applying t-SNE.



■ **Figure 3.8** Performance estimation plot.

The user can view the preprocessed plot of the currently selected spectrum. In addition, the user has the option to view a raw plot and metadata of the current spectrum by clicking the button under the preprocessed plot.



■ **Figure 3.9** Plot of preprocessed spectrum.

The Plotly [43] library is used to render interactive plots. A specific area of a plot can be zoomed in on by drawing a box, or the plot can be panned. To switch between these two modes, the user needs to hover over the plot; several buttons will then appear in the top-right corner, and the user can click the desired one. To restore the initial view, the user can double-click on the plot or click the “reset axes” button in the top-right corner. By hovering over a data point, the user can see its value. In addition, the user can download an image of the plot. See code listing 3.3 to view an example of rendering a plot.

3.2.4.3 File system

On this page, the user can interact with the platform’s storage, which is displayed as a file system. Initially, files are displayed along with their size in bytes and the date they were last modified, followed by a list of directories. Directories are distinguishable from files by being underlined and highlighted in blue. To open a directory, the user needs to click on its name. The user can also create new directories and upload files. If a file with an identical name as

```
import React from 'react';
import Plot from "react-plotly.js";

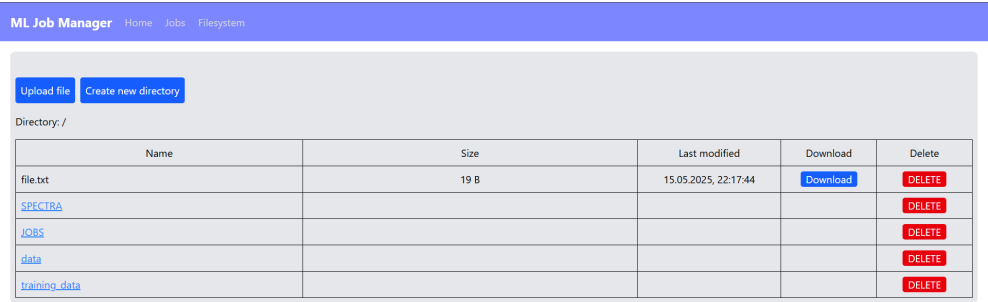
interface PlotProps {
  title: string;
  waves: number[];
  fluxes: number[];
}

const SpectrumPlot: React.FC<PlotProps> = ({title, waves,
↪ fluxes}) => {
  return (
    <div>
      <Plot
        data={[{ x: waves, y: fluxes}]}
        layout={{
          title: {text: title},
          xaxis: {
            title: {text: 'Wavelengths'},
            zeroline: false
          },
          yaxis: {
            title: {text: 'Fluxes'},
            zeroline: false
          },
          autosize: true
        }}
        style={{width: '100%', height: '100%'}}
      />
    </div>
  );
};

export default SpectrumPlot;
```

■ **Code listing 3.3** Example of plot rendering component.

the uploaded file exists in the directory, this file will be overwritten. Deleting a directory removes all the contained files and subdirectories as well.



■ **Figure 3.10** Displaying file system.

3.3 Workflow

A typical user workflow proceeds as follows:

1. The user opens the preprocessing job creation page, fills the form, submits it, and starts it. The system runs a preprocessing job and saves the result.

```
{
  "wave_start_point": 5000,
  "wave_end_point": 6500,
  "wave_point_count": 150,
  "data_dir_path": "/B6001"
}
```

■ **Code listing 3.4** Example of preprocessing job’s configuration file.

2. Once the preprocessing job is completed, the user navigates to the active learning job creation page. The user fills out the form and specifies in the configuration that it is the zero iteration, also sets the path to the preprocessing job result to “**pool_data_path**” parameter. Then, the user submits and starts the job. After the zero iteration is completed, only the ORACLE spectra are displayed for the labeling. After completing labeling, the user starts the first iteration using either the configuration file created by zero iteration or by completing it manually.
3. After the regular iteration of the active learning job completes successfully, the user estimates the performance of the iteration, views the plot of performance, and if satisfied, ends the work, otherwise starts labeling the ORACLE spectra and starts the next iteration.

```
{
  "iteration": 0,
  "pool_data_path": "/pool_data_dir/pool_data.h5",
  "classes": [
    "other",
    "single peak",
    "double peak"
  ],
  "candidate_classes": [
    "single peak",
    "double peak"
  ],
  "oracle_batch_size": 100
}
```

■ **Code listing 3.5** Example of active learning job's configuration file for zero iteration.

```
{
  "iteration": 1,
  "pool_data_path": "/pool_data_dir/pool_data.h5",
  "training_data_path":
    ↪ "training_data_dir/training_data_path.h5"
  "classes": [
    "other",
    "single peak",
    "double peak"
  ],
  "candidate_classes": [
    "single peak",
    "double peak"
  ],
  "oracle_batch_size": 30,
  "perf_est_size": 10
}
```

■ **Code listing 3.6** Example of active learning job's configuration file for first and subsequent iterations.

3.3.1 Launching

The front-end is containerized and orchestrated using Docker Compose [44]. By executing **docker compose up** command the front-end will be launched

without further setup or configuration.

```
FROM node:20-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 10300
CMD ["npm", "run", "dev"]
```

■ **Code listing 3.7** Front-end Dockerfile.

```
services:
  ml-job-client:
    image: ml-job-manager-client
    container_name: ml-job-manager-client
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "10300:10300"
```

■ **Code listing 3.8** Front-end Docker Compose file.

Chapter 4

Discussion

4.1 Performance

The preprocessing and active deep learning modules performance depends on their configuration, since their core logic remains unchanged and only their parameters are now configurable.

The preprocessing module, where the wavelength range was set from 5000 Å to 8500 Å with 1000 uniform points, processed 2431 LAMOST DR2 FITS files completed in 28.05 s. Active deep learning module was launched on **NVIDIA GeForce GTX980** GPU with training data consisting of 1000 samples and pool data containing 2431 samples, where each spectrum has wavelengths ranging from 5000 Å to 8500 Å and 1000 flux counts, and it took 44.23 seconds.

When creating a new directory, the front-end does not re-fetch the list containing all files and directories. Instead, after receiving a successful response, the newly created directory is added to the existing list in the web interface. The same logic applies to file uploading, file and directory deletion, and committing spectrum labels and comments. This approach reduces the load on the infrastructure.

4.2 Future improvements

Some improvements written below requires corresponding back-end extension.

- Improve user interface design. Make it more user-friendly, for example, replace the JSON file configuration with a form-based input method.
- After labeling spectra on the active learning job details page, enable the user to start the next iteration from that page.
- Integrating user authentication and logging. Implementing access control based on roles.

- Uploading multiples files. Deleting multiple files and directories.
- Using custom model dialog instead of browser synchronous dialog, as it blocks the whole page until the user responds.
- Enabling to work with the other astronomical data.
- Adding new types of jobs for processing astronomical data.
- Adding new analysis modules for processing astronomical data.
- Instead of checking every 10 seconds the job's status by sending a request to the server, use a websocket.

Conclusion

The primary goal of the thesis was to design, implement, and integrate analysis modules - preprocessing, active deep learning, and dimensionality reduction. All three modules can be launched from the terminal. Moreover, the front-end was developed, which enables the launch of preprocessing and active deep learning modules through the web interface.

The aim of the thesis and all requirements were fulfilled. The preprocessing module processes the raw spectra, enabling the application of the active deep learning module. This module can help identify interesting or unusual spectra. Meanwhile, the dimensionality reduction module improves the comprehension of the data. Through the web interface, the user can conveniently and easily launch these modules.

Appendix A

Launching the programs

For launching modules from terminal go to modules directory and run following commands:

```
python -m venv env
source env/bin/activate
pip install -r requirements.txt
env/bin/python /directory_name/file_name config_path
↪ result_directory
```

■ **Code listing A.1** Example of module launching.

Main file of each module has the name `job_*.py`

For running front-end and infrastructure(back-end), go to `ml-job-manager-client` and `ml-job-manager` directories and run "docker compose up" from terminal. When building a container for the back-end, it will take 65-75 GB of disk space. The front-end will run on `localhost:10300`, the back-end API will run on `localhost:10000`. After starting the container `/ml-job-manager` directory is created, this directory will be used for storing the data. Inside `/ml-job-manager` directory, `SPECTRA` directory is created, which will contain LAMOST DR2 FITS files. Insert in `/ml-job-manager/SPECTRA` directory from `B6001.zip` and `F5902.zip` from the attachments. Do not change the name of these directories. The path and directory name can be changed in the `.env` file inside the `ml-job-manager` directory.

Bibliography

1. *Machine Learning / Center for Astrophysics / Harvard & Smithsonian* [online]. 2024. Available also from: <https://pweb.cfa.harvard.edu/research/topic/machine-learning>. [Accessed 2025-05-13].
2. KOZA, Jakub. *Design and implementation of a distributed platform for data mining of big astronomical spectra archives* [online]. 2017. Available also from: <https://dspace.cvut.cz/bitstream/handle/10467/63145/F8-BP-2015-Koza-Jakub-thesis.pdf?sequence=2&isAllowed=y>. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology. [Accessed 2025-05-12].
3. KOZA, Jakub. *Interactive Cloud-Based Platform for Parallelized Machine Learning of Astronomical Big Data* [online]. 2017. Available also from: <https://dspace.cvut.cz/bitstream/handle/10467/69141/F8-DP-2017-Koza-Jakub-thesis.pdf?sequence=1&isAllowed=y>. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology. [Accessed 2025-05-12].
4. PODSZTAVEK, Ondřej. *active-cnn/active_cnn at master · podondra/active-cnn — github.com* [online]. 2019. Available also from: https://github.com/podondra/active-cnn/tree/master/active_cnn. [Accessed 2025-04-09].
5. PODSZTAVEK, Ondřej. *Deep Learning in Large Astronomical Spectra Archives* [online]. 2017. Available also from: <https://dspace.cvut.cz/bitstream/handle/10467/69666/F8-BP-2017-Podsztavek-Ondrej-thesis.pdf?sequence=1&isAllowed=y>. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology. [Accessed 2025-05-12].
6. ŠKODA, P.; PODSZTAVEK, O.; TVRDÍK, P. Active deep learning method for the discovery of objects of interest in large spectroscopic surveys. *Astronomy & Astrophysics*. 2020, vol. 643, A122. Available from DOI: 10.1051/0004-6361/201936090. Published online: 2020-11-11.

7. MASSEY, Philip; HANSON, Margaret M. Astronomical spectroscopy. *arXiv preprint arXiv:1010.5270*. 2010.
8. *Spectra - Introduction* [online]. 2013. Available also from: <https://imagine.gsfc.nasa.gov/science/toolbox/spectra1.html>. [Accessed 2025-05-07].
9. *FITS Primer* [online]. 2014. Available also from: https://fits.gsfc.nasa.gov/fits_primer.html. [Accessed 2025-05-10].
10. *Hierarchical Data Formats - What is HDF5? / NSF NEON / Open Data to Understand our Ecosystems* [online]. 2025. Available also from: <https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5>. [Accessed 2025-05-10].
11. *LAMOST Data Release Two - LAMOST DR2 — dr2.lamost.org* [online]. 2015. Available also from: <https://dr2.lamost.org/doc/data-production-description>. [Accessed 2025-04-02].
12. LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. *nature*. 2015, vol. 521, no. 7553, pp. 436–444. Available from DOI: 10.1038/nature14539.
13. SETTLES, Burr. *Active Learning Literature Survey*. 2009. Computer Sciences Technical Report, 1648. University of Wisconsin–Madison.
14. VAN DER MAATEN, Laurens; HINTON, Geoffrey. Visualizing data using t-SNE. *Journal of machine learning research*. 2008, vol. 9, no. 11.
15. *Project Jupyter / Home* [online]. [N.d.]. Available also from: <https://jupyter.org/>. [Accessed 2025-05-16].
16. *W3C XHTML2 Working Group Home Page — w3.org* [online]. 2013. Available also from: <https://www.w3.org/Markup/>. [Accessed 2025-04-14].
17. BONNER, Anne. *What is a CSV file: A comprehensive guide / Flatfile*. 2020. Available also from: <https://flatfile.com/blog/what-is-a-csv-file-guide-to-uses-and-benefits/>. [Online; accessed 2025-05-14].
18. *Elasticsearch: The Official Distributed Search & Analytics Engine / Elastic*. [N.d.]. Available also from: <https://www.elastic.co/elasticsearch>. [Online; accessed 2025-05-14].
19. MAZEL, Tomáš. *Cloud-Based Platform for Active Learning of Astronomical Spectra* [online]. 2020. Available also from: <https://dspace.cvut.cz/bitstream/handle/10467/88254/F8-BP-2020-Mazel-Tomas-the-sis.pdf?sequence=-1&isAllowed=y>. Bachelor’s thesis, Czech Technical University in Prague, Faculty of Information Technology. [Accessed 2025-05-12].

20. *vodev/vocloud-active-learning: Active learning worker for vo-cloud*. 2020. Available also from: <https://github.com/vodev/vocloud-active-learning>. [Online; accessed 2025-05-14].
21. SAUZAY, Armand. *Which Data Format to Use For Your Big Data Project? | Towards Data Science*. 2023. Available also from: <https://towardsdatascience.com/which-data-format-to-use-for-your-big-data-project-837a48d3661d/>. [Online; accessed 2025-05-14].
22. BECKER, Pablo; TEBES, Guido; PEPPINO, Denis; OLSINA SANTOS, Luis Antonio. Applying an improving strategy that embeds functional and non-functional requirements concepts. *Journal of Computer Science & Technology*. 2019, vol. 19.
23. MARTÍN ABADI; ASHISH AGARWAL; PAUL BARHAM; EUGENE BREVDO; ZHIFENG CHEN; CRAIG CITRO; GREG S. CORRADO; ANDY DAVIS; JEFFREY DEAN; MATTHIEU DEVIN; SANJAY GHEMAWAT; IAN GOODFELLOW; ANDREW HARP; GEOFFREY IRVING; MICHAEL ISARD; JIA, Yangqing; RAFAL JOZEFOWICZ; LUKASZ KAISER; MANJUNATH KUDLUR; JOSH LEVENBERG; DANDELION MANÉ; RAJAT MONGA; SHERRY MOORE; DEREK MURRAY; CHRIS OLAH; MIKE SCHUSTER; JONATHON SHLENS; BENOIT STEINER; ILYA SUTSKEVER; KUNAL TALWAR; PAUL TUCKER; VINCENT VANHOUCKE; VIJAY VASUDEVAN; FERNANDA VIÉGAS; ORIOL VINYALS; PETE WARDEN; MARTIN WATTENBERG; MARTIN WICKE; YUAN YU; XIAOQIANG ZHENG. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Available also from: <https://www.tensorflow.org/>. Software available from tensorflow.org.
24. *tf.keras.Sequential | TensorFlow v2.16.1* [online]. 2024. Available also from: https://www.tensorflow.org/api_docs/python/tf/keras/Sequential. [Accessed 2025-05-04].
25. *tf.keras.callbacks.EarlyStopping | TensorFlow v2.16.1* [online]. 2024. Available also from: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping. [Accessed 2025-05-04].
26. HARRIS, Charles R.; MILLMAN, K. Jarrod; WALT, Stéfan J. van der; GOMMERS, Ralf; VIRTANEN, Pauli; COUNAPEAU, David; WIESER, Eric; TAYLOR, Julian; BERG, Sebastian; SMITH, Nathaniel J.; KERN, Robert; PICUS, Matti; HOYER, Stephan; KERKWIJK, Marten H. van; BRETT, Matthew; HALDANE, Allan; RÍO, Jaime Fernández del; WIEBE, Mark; PETERSON, Pearu; GÉRARD-MARCHANT, Pierre; SHEPPARD, Kevin; REDDY, Tyler; WECKESSER, Warren; ABBASI, Hameer; GOHLKE, Christoph; OLIPHANT, Travis E. Array programming with NumPy. *Nature*. 2020, vol. 585, no. 7825, pp. 357–362. Available from DOI: 10.1038/s41586-020-2649-2.

27. *What is NumPy? — NumPy v2.2 Manual* [online]. 2024. Available also from: <https://numpy.org/doc/2.2/user/whatiscnumpy.html>. [Accessed 2025-04-15].
28. PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, vol. 12, pp. 2825–2830.
29. *scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation* [online]. [N.d.]. Available also from: <https://scikit-learn.org/stable/>. [Accessed 2025-04-15].
30. *What is Angular? • Angular* [online]. 2025. Available also from: <https://angular.dev/overview>. [Accessed 2025-05-04].
31. WAITE, Robin. *Angular Framework: Advantages and Disadvantages Elaborated* [online]. 2024. Available also from: <https://www.robinwaite.com/blog/pros-and-cons-of-angular-framework-you-need-to-know>. [Accessed 2025-05-04].
32. *Introduction to the DOM - Web APIs / MDN* [online]. 2025. Available also from: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. [Accessed 2025-05-16].
33. *React* [online]. [N.d.]. Available also from: <https://react.dev/>. [Accessed 2025-04-18].
34. *Next.js vs. React: The difference and which framework to choose / Contentful* [online]. [N.d.]. Available also from: <https://www.contentful.com/blog/next-js-vs-react/>. [Accessed 2025-05-02].
35. JONNA, Vaishnavi. *Advantages and Disadvantages of React js - ellow.io* [online]. 2024. Available also from: <https://ellow.io/advantages-and-disadvantages-of-react-js/>. [Accessed 2025-05-08].
36. TRUE. *Next.js by Vercel - The React Framework*. 2025. Available also from: <https://nextjs.org/>. [Online; accessed 2025-05-04].
37. EMADAMERHO-ATORI, Nefe. *Next.js Pros and Cons compared* [online]. 2024. Available also from: <https://www.altexsoft.com/blog/nextjs-pros-and-cons/>. [Accessed 2025-05-08].
38. *Getting Started / Vite* [online]. [N.d.]. Available also from: <https://vite.dev/guide/>. [Accessed 2025-05-04].
39. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. [Online]. 2025. Available also from: <https://tailwindcss.com/>. [Online; accessed 2025-05-04].

40. *TypeScript: JavaScript With Syntax For Types*. [Online]. [N.d.]. Available also from: <https://www.typescriptlang.org/>. [Accessed 2025-05-15].
41. IBM. *What Is a REST API (RESTful API)? / IBM* [online]. 2025. Available also from: <https://www.ibm.com/think/topics/rest-apis>. [Accessed 2025-05-16].
42. *An overview of HTTP - HTTP / MDN* [online]. 2025. Available also from: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview>. [Accessed 2025-05-16].
43. *React plotly.js in JavaScript* [online]. 2025. Available also from: <https://plotly.com/javascript/react/>. [Accessed 2025-05-15].
44. *Docker Compose / Docker Docs* [online]. 2025. Available also from: <https://docs.docker.com/compose/>. [Accessed 2025-05-16].

Contents of the attachment

```
/
├── readme.txt ..... brief description of content
├── src
│   ├── modules ..... analysis modules
│   ├── ml-job-client ..... front-end source code
│   ├── ml-job-manager ..... infrastructure with integrated modules
│   └── thesis ..... source form of thesis in LATEX format
├── text ..... thesis text
│   └── thesis.pdf ..... thesis text in PDF format
├── datasets
│   ├── training_data.h5 ..... training data
│   └── pool_data.h5 ..... pool data
├── B6001 ..... directory containing some LAMOST DR2 FITS files
└── F5902 ..... directory containing some LAMOST DR2 FITS files
```