CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Deep Learning in Large Astronomical Spectra Archives
Student:	Ondřej Podsztavek
Supervisor:	RNDr. Petr Škoda, CSc.
Study Programme:	Informatics
Study Branch:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	Until the end of winter semester 2018/19

Instructions

The goal is the identification of emission-line spectra in the LAMOST spectral survey archive (containing millions of spectra) using deep neural networks (NNs) trained on samples of selected objects from the Ondřejov 2m telescope.

1) Make a survey of current state of deep learning, taking into account the need of massive parallelisation on clusters and/or GPUs to process LAMOST archives.
2) Collect and classify the prototypes of emission line stars spectra in the Ondřejov archive.
3) Apply domain adaptation from Ondřejov to LAMOST spectral resolution using Gaussian convolution.
4) Investigate the structure of data space in both archives using dimensional reduction and use this to select a proper deep NN architecture.
5) Train the deep NN on data obtained in 3) using labels from 2) .
6) Prepare visualisation of resulting candidates of the target class.
7) Discuss the precision, performance and scalability in various configurations of the NN and suggest future improvements.

References

Will be provided by the supervisor.

doc. Ing. Jan Janoušek, Ph.D.	prof. Ing. Pavel Tvrdík, CSc.
Head of Department	Dean

Prague February 21, 2017

Bachelor's Thesis

Czech
Technical
University
in Prague

F8

Faculty of Information Technology
Department of Theoretical Computer Science

Deep Learning in Large Astronomical Spectra Archives

Ondřej Podsztavek
Branch of Study: Computer Science

May 2017
Supervisor: RNDr. Petr Škoda, CSc.

Acknowledgement / Declaration

This work would have been impossible without the support of Petr Škoda, his domain expertise and enthusiasm. I would like to thank to my family for their extraordinary support during my bachelor's degree. This research was supported by the grant COST LD-15113 of the Ministry of Education Youth and Sports of the Czech Republic.

Abstrakt / Abstract

Velké astronomické archívy, jako například spektrální archív LAMOST, obsahují řadu skrytých informací. Hluboké učení je jednou z nejpopulárnějších dnes používaných metod pro získávání znalostí z tohoto druhu dat. Tato práce popisuje proces hledání spekter s emisními čarami v archívu LAMOST za použití hluboké konvoluční neuronové sítě naučené na datech z ondřejovského 2m teleskopu. Práce popisuje několik metod jako je předzpracování spekter, doménová adaptace ondřejovských dat na rozlišení archívu LAMOST, redukce dimenzionality, návrh a učení dvou neuronových sítí. V závěru práce je diskuze objevených objektů se zajímavou fyzikální podstatou, které vyžadují další detailní analýzu.

Klíčová slova: hluboké učení, neuronové sítě, redukce dimenzionality, doménová adaptace, astroinformatika, astronomie, LAMOST, TensorFlow

Překlad titulu: Hluboké učení ve velkých archívech astronomických spekter

Large astronomical archives, as for example LAMOST spectral archive, contain plenty of hidden information. Deep learning is currently very popular method used to gain knowledge from this kind of data. This work shows the process of finding emission-line spectra in LAMOST archive using deep convolutional neural network trained on data from Ondřejov 2m telescope. Overview of several techniques as spectra preprocessing, domain adaptation of Ondřejov data to LAMOST resolution, dimensionality reduction, architecture and training of two deep neural networks are presented. Finally, discovered objects with interesting physical nature deserving further detailed analysis are discussed.

Keywords: deep learning, neural networks, dimensionality reduction, domain adaptation, astroinformatics, astronomy, LAMOST, TensorFlow

Contents

Tables / Figures

Chapter 1
Introduction

Astronomy and science in general are transformed by exponential growth of data which provides huge opportunities for discovery. Modern large telescopes like large mosaics of CCD chips are producing terabytes of raw data per night. For example the world largest spectrograph of LAMOST telescope is acquiring 4 000 spectra per single exposure. Astronomy is facing an avalanche of data that can be process only with sophisticated and innovative approaches.

Recent advance in deep neural networks have brought breakthroughs in processing images, video, speech and audio. Deep learning is able to discover intricate structure in large datasets. Thus deep networks should also help to make new discoveries in astronomy.

The goal of this work is to identify emission-line spectra in the LAMOST spectral survey archive using deep neural network which is trained on spectra from Ondřejov archive.

This work starts with introduction to spectral data in chapter 2. Chapters 3 and 4 introduce basics of machine, deep and transfer learning. In chapter 5 is survey of currently available Python frameworks for deep learning. Chapter 6 describes the creation and properties of Ondřejov dataset which is used for network training. Chapter 7 is about preprocessing methods applied to the data. Visualizations of Ondřejov and LAMOST data space are shown in chapter 8. The last chapter 9 presents architecture of the deep neural network, its training and classification results of spectra from LAMOST.

Chapter 2
Spectral Data

This thesis is concerned with classification of star's spectra. Aim of this chapter is to describe this kind of data. The following sections introduce spectroscopy, its related concepts, Ondřejov and LAMOST astronomical archives of spectral data.

2.1 Astronomical Spectroscopy

Spectroscopy in astronomy is the study of celestial objects' spectra. The different wavelengths of electromagnetic radiation are spread out into a spectrum. This information is then used to derive chemical composition, temperature, distance, relative motion and much more. [1]

2.2 Electromagnetic Radiation

Electromagnetic radiation is pair of electric and magnetic fields that propagate together at the speed of light ($c = 299\,792\,458\,\mathrm{ms}^{-1}$). Visible light, radio waves, X-rays and gamma rays are examples of electromagnetic radiation. Electromagnetic spectrum is collective term for known range of electromagnetic radiation. The electric and magnetic field oscillates and produces electromagnetic waves. [2]

The electromagnetic wave is described in terms of frequency or wavelength:

- Frequency (f) is the number of waves per second and its unit is Hertz.
- Wavelength (λ) is the distance between successive crests or troughs in the wave and it is measured is meters or astronomy usually uses Ångströms ($1\,\text{Å} = 10^{-10}\,\mathrm{m}$).

Frequency and wavelength are related by wave equation:

$$c = \lambda f \tag{2.1}$$

2.3 Black Body Radiation

A black body is a hypothetical object which is a perfect absorber and emitter of radiation over all wavelengths. The spectral flux distribution of black body's thermal energy depends on its temperature (see figure 2.1). Stars are often modeled as black bodies in astronomy. Their spectrum approximates the black body spectrum. [2]

2.4 Spectrum Continuum

Each spectrum is characterized by the continuum and the spectral lines. The continuum is generally smoothly varying spectrum of light emitted by a star while spectral lines are peaks reaching out from a continuum. [3]

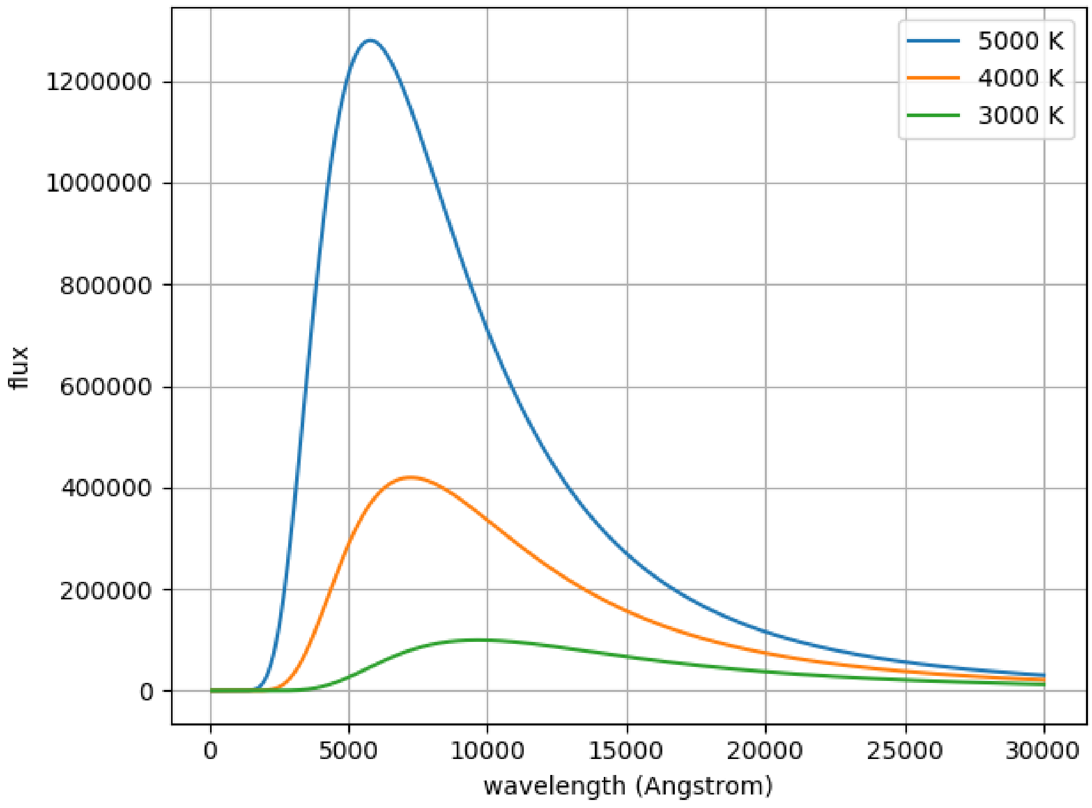

Figure 2.1. Black body radiation curves of 3 000, 4 000 and 5 000 Kelvins hot stars.

In order to avoid variations between different observations a continuum is usually normalized to serve as reference point to spectral lines (see figure 2.2). A smooth curve is fit to the continuum and then the spectrum is divided by it. This sets the continuum everywhere to value 1 and allows to measure spectral lines in consistent way.

2.5 Spectral Lines

Spectral lines can be used to identify the chemical composition of stars. If a light from a star is separated with a prism its spectrum of colors is crossed with discrete lines. This can be also visualized as flux of particular wavelengths. **Flux** is the total amount of energy that crosses a unit area per unit time.

There are two types of spectral lines:

- emission and
- absorption lines.

Emission line occurs when atom in a higher energy level (excited state) return to lower energy level and releases energy. According to quantum theory every atom has a unique set of energy levels. Therefore, the atom can emit electromagnetic radiation of particular wavelengths equal to the difference between the energy levels. Energy and wavelength, frequency are related through Planck-Einstein relation:

$$E = hf, \quad E = \frac{hc}{\lambda} \tag{2.2}$$

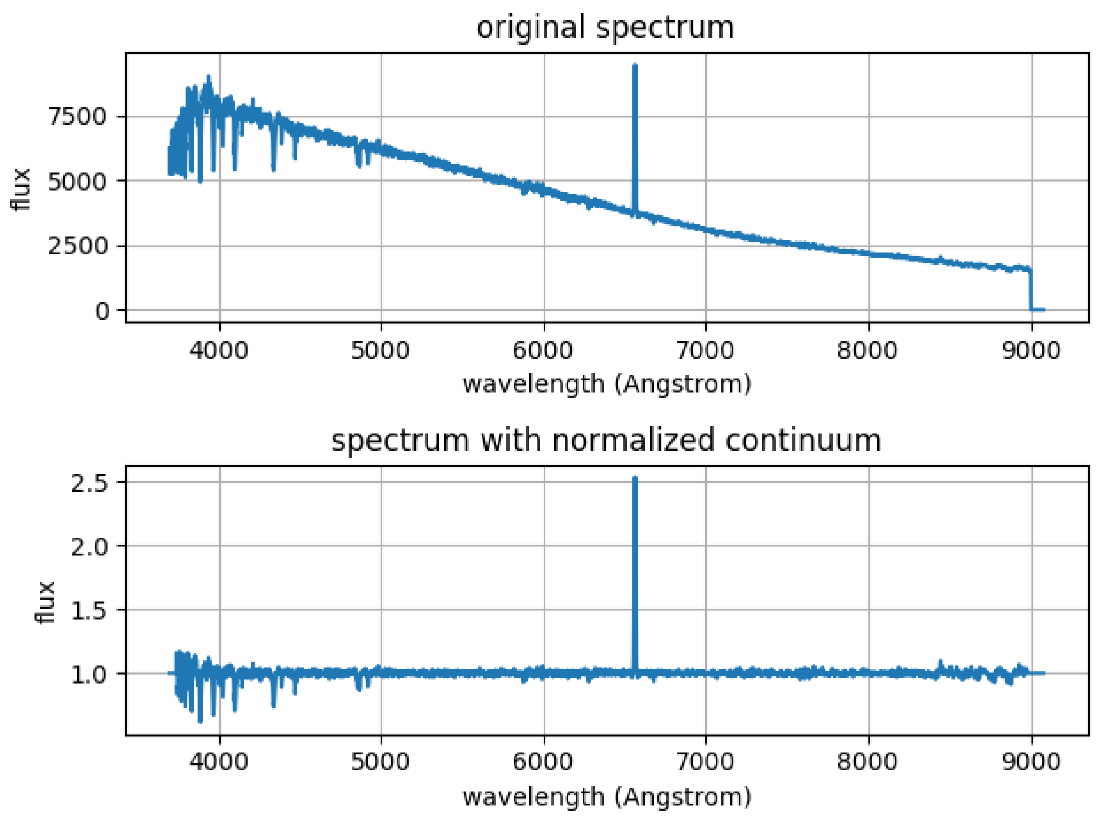

Figure 2.2. Comparison between raw spectrum and corresponding spectrum with normalized continuum.

where $h = 6.62607004 \cdot 10^{-34}\,\mathrm{J \cdot s}$ is Planck constant and c is the speed of light. On a graph of wavelengths fluxes emission lines appear as peaks above the continuum level.

Absorption lines are opposite of emissions. They will appear when there is an absorbing material between the source and the observer. The material could be outer layers of a stars or interstellar gas. Atoms will absorb specific energies from the electromagnetic spectrum specifically to atom's energy levels. On graph these absorption features are show below the level of a star's black body continuum spectrum.

2.5.1 Balmer Lines

The Balmer lines or Balmer series is the name of spectral lines of hydrogen atom that result from electron transitions between second energy level and higher levels. There are four transitions that are in visible wavelength. These are named Hα, Hβ, Hγ and Hδ. Because hydrogen is most abundant element Balmer lines are a commonly observed features in spectroscopy. [2]

Hα is deep-red visible spectral line. Its wavelength in air is 6 562.8 Å and in vacuum is 6 564.6 Å. This spectral line is created when electron moves between the second and third energy level of hydrogen atom.

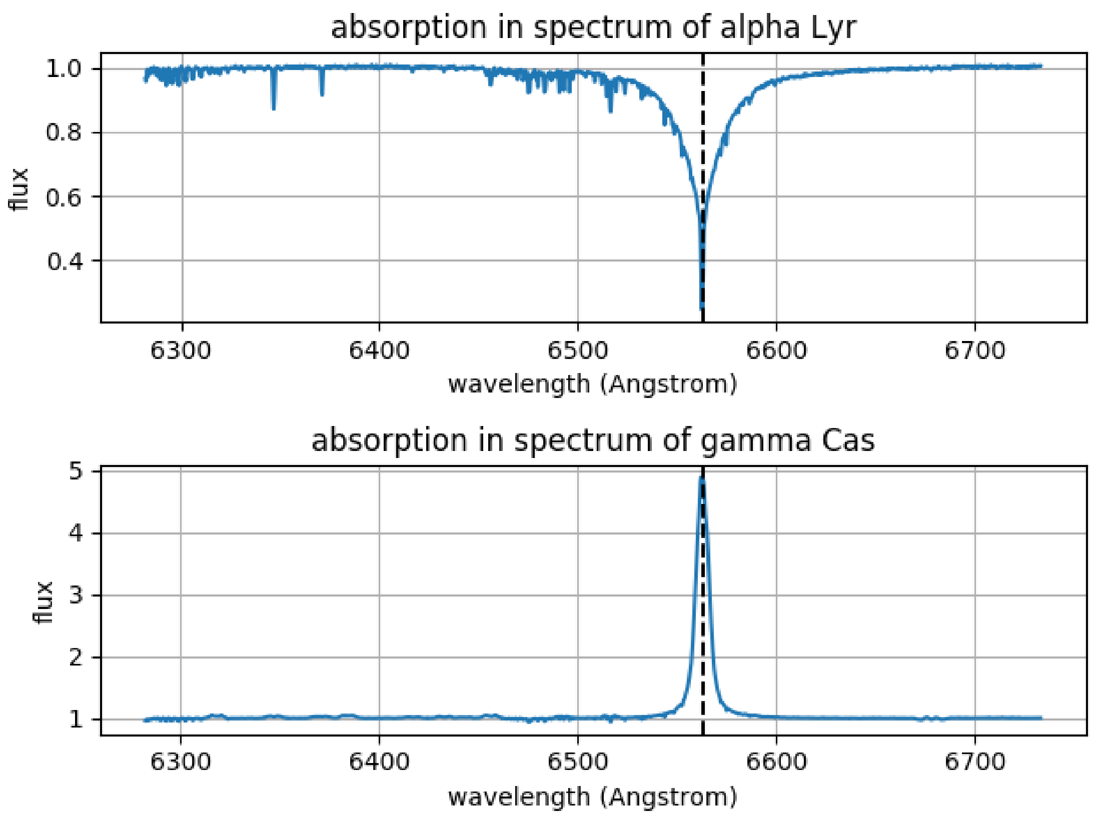

Figure 2.3. Spectra of gamma Cas and alpha Lyr with emission and absorption line in Hα respectively.

2.6 Be Stars

Be stars are **non-supergiant B stars**[1] with spectrum that has or had one or more **Balmer lines in emission at some time**. Although this definition is not precise it is sufficient for purpose of this work. [5]

It is believed that a Be star consists of fast rotating star and star disk which may cause the emissions in Balmer lines. Absorptions and emissions in electromagnetic spectrum of Be star can periodically vary.

2.7 Ondřejov Archive

Archive of Ondřejov observatory spectral data is available at ASU CAS Data Center[2]. It currently contains about 17 000 spectra [6]. Because this archive is **specialized on Be stars observations** it contains approximately 13 000 spectra with Hα spectral line. The data were obtained with Ondřejov Perek 2 m telescope, coudè camera with focus 700 mm and two different detectors changed over time. The spectrograph's spectral resolving power is about 13 000 in Hα [7].

Spectral resolving power of a spectrograph is defined as:

[1] Stars from spectral type B are hydrogen burning stars which have 2 to 16 times the mass of the Sun and surface temperatures between 10 000 and 30 000 K. [4]

[2] `http://voarchive.asu.cas.cz/`

$$R = \frac{\lambda}{\Delta\lambda} \tag{2.3}$$

where $\Delta\lambda$ is **spectrum resolution** which stands for the smallest difference in wavelengths that can be distinguished at a wavelength of λ.

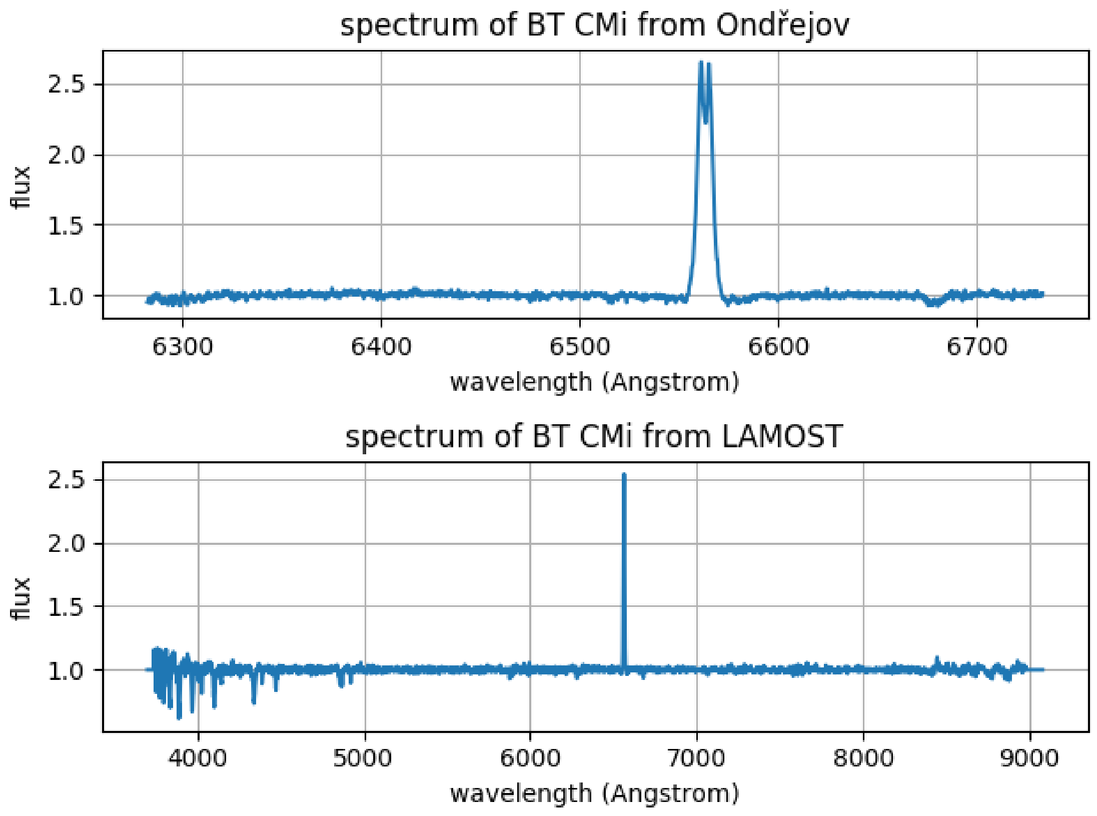

Figure 2.4. Spectra of BT CMi star from Ondřejov and LAMOST archive.

2.8 LAMOST Archive

The Large Sky Area Multi-Object Fiber Spectroscopic Telescope (LAMOST) is telescope located in China with spectral resolving power 500, 1 000 or 1 500. LAMOST archive contains more than 7 millions of spectra. The latest observations are available in data release 5 Q1[1]. [8]

survey	total spectra	stars
pilot	958 944	812 911
first year	1 701 669	1 529 958
second year	1 648 485	1 501 002
third year	1 659 028	1 511 032
forth year	1 713 059	1 543 415
total	7 681 185	6 898 318

Table 2.1. Statistics of LAMOST data releases according to [9].

[1] http://dr5.lamost.org/

2.9 FITS File Format

Flexible Image Transport System is data format used within astronomy for transporting, analyzing, archiving scientific data files. It is designed to store datasets consisting of multidimensional arrays and two dimensional tables. [10]

A FITS file is comprised of segments called Header/Data Units (HDUs). The first HDU is called the primary HDU. The primary data array can contain a 1–999 dimensional array of numbers. A typical primary array could contain a **1 dimensional spectrum**, a 2 dimensional image, a 3 dimensional data cube.

Any number of additional HDUs may follow the primary array. These HDUs are referred as extensions. There are three types of standard extensions currently defined:

- Image Extension
- ASCII Table Extension
- Binary Table Extension

Every HDU consists of an ASCII formatted header unit and data unit.

Each header unit contains a sequence of fixed-length 80 character long keyword record which has form:

```
KEYNAME = value / comment string
```

Non-printing ASCII character such as tabs, carriage-returns, line-feeds are not allowed anywhere in the header unit.

Note that the data unit is not required. The image pixels in primary array or an image extension may have one of 5 supported data types:

- 8-bit (unsigned) integer bytes
- 16-bit (signed) integer bytes
- 32-bit (signed) integer bytes
- 32-bit single precision floating point real numbers
- 64-bit double precision floating point real numbers

The other 2 standard extensions, ASCII tables and binary tables, contain tabular information organized into rows and columns. Binary tables are more compact and are faster to read and write then ASCII tables.

All the entries within a column of a table have the same datatype. The allowed data formats for an ASCII table column are integer, single and double precision floating point value, character string. Binary table also support logical, bit and complex data formats.

Chapter 3
Machine Learning

Machine learning is subfield of artificial intelligence [11]. Algorithms in machine learning allow computer programs to learn from data [12]. Samuel Arthur proposed this famous statement about machine learning in [13]: "*Programming computers to learn from experience should eventually eliminate the need for much of this programming effort.*" Formal and widely used definition is provided by Tom Mitchell in [14]: "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*".

Therefore, instead of hard coding the desired model directly into program, data are fed into machine learning algorithm which develops its own model. This is called **data-driven approach** since it depends on providing the algorithm with vast amount of data. [15]

For example considering the image classification program it would be challenging to hard code into program how general dog looks. Machine learning algorithm in combination with data-driven approach would take large collection of dog's images, feed them into program and let it develop its own notion of what a dog looks like.

Commonly, machine learning is divided into **supervised** and **unsupervised** learning. In supervised learning, there is a dataset and its corresponding output values. Supervised algorithms are used for solving **classification** or **regression** tasks. In classification inputs are assigned from a set of discrete values. On contrary, regression predicts outputs from continuous range. Unsupervised learning allows to solve problems where is no or little idea about the result. [12]

3.1 Deep Learning

Conventional machine learning techniques were limited in ability to process raw natural data. Machine learning algorithm required careful design of feature extractor that transformed the raw data into suitable feature vectors from which the learning subsystem could classify patterns in the input.

Deep learning methods are **representation learning methods** with multiple levels of representation obtained by composing simple but non-linear modules. With the composition of enough such transformation higher layers of representation amplify aspects of the input that can be important for discrimination. The key is that these layers of features are not designed by humans but learned from data. [16]

3.2 Feedforward Neural Networks

Many applications of deep learning use feedforward neural network architectures. Schema of feedforward neural network in in figure 3.1. To go from one layer to the next a set of units compute a weighted sum of their inputs from previous layer and

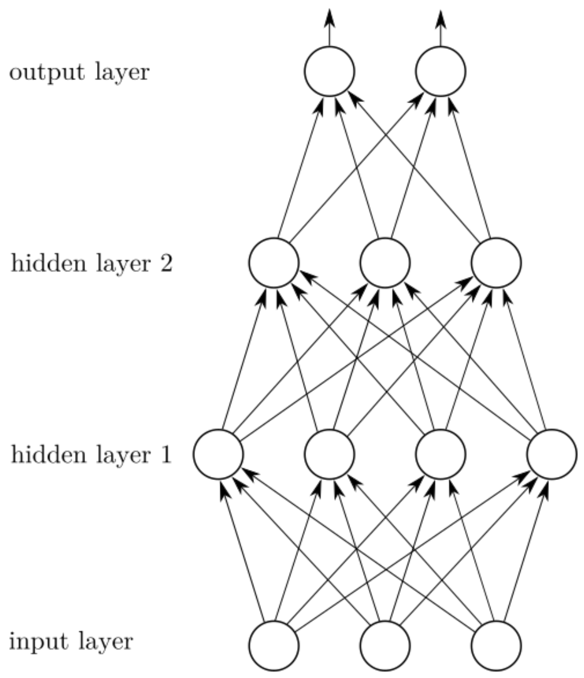

Figure 3.1. Feedforward neural network with 4 **fully connected layers**. The **input layer** has 3 units followed 2 **hidden layers** with 4 and 3 units. **Output layer** contains 2 units.

apply non-linear function to the result. Units that are not in the input or output layer are called hidden units. These hidden layers of a multi layer neural network can distort the input space to make the classes linearly separable.

3.3 Convolutional Networks

Convolutional neural networks (CNNs, ConvNets) are one particular type of deep feed-forward network that is much easier to train and generalize much better than fully connected networks. [16]

These networks are designed to process data in form of multiple arrays, for instance **1 dimensional signals** or 2 dimensional images. They take advantage of four properties:

- local connections,
- shared weights,
- pooling,
- use of many layers.

Typical convolutional neural network is structured as series of stages. First stages are composed of convolutional layers and pooling layers. Units in convolutional layer maps to next layer through several **filters** that are convoluted with inputs from previous layer. These results are passed through a non-linearity. Nowadays, the most widely used non-linearity is **rectified linear unit** (ReLU):

$$f(z) = \max(z, 0) \tag{3.1}$$

The reasons for this architecture are that local groups of values are highly correlated and motifs can appear in any part of image or signal. Mathematically, the filtering operation is discrete convolution, hence the name.

Pooling layers are used to merge semantically similar features into one. This layer usually computes maximum of a local patch of units. Thereby, they reduce the dimensions of the representation and invariance to small shifts. [16]

Chapter 4
Transfer Learning

A common assumption in machine learning is that the training and data faced in deployment must be in same feature space and have the same probability distribution (see diagram 4.1). For example, this thesis is interested in classification of LAMOST archive but its training set from Ondřejov spectrograph. In such cases as much knowledge as possible need to be transfer to maximally improve performance of final classifier on target data. [17]

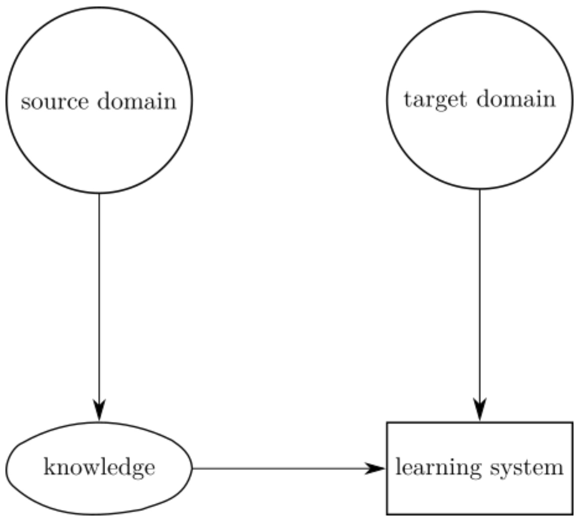

Figure 4.1. Diagram of transfer learning process. Inspired by [17].

Formal definition of transfer learning is provided in [17]. **Domain** $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a probability distribution $P(X)$ where $X = \{x_1, ..., x_n\} \in \mathcal{X}$ is a learning sample. Given a specific $\mathcal{D} = \{\mathcal{X}, P(X)\}$ and a learning **task** $\mathcal{T} = \{\mathcal{Y}, f\}$ where $\mathcal{Y}$ is a label space and f is a predictive function. This predictive function f is learned from training set which consist of pairs $\{x_i \in X, y_i \in \mathcal{Y}\}$. From probabilistic perspective f is $P(y \mid x)$.

Therefore, having a source domain $\mathcal{D}_S$ and a task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a task $\mathcal{T}_T$, **transfer learning** aims to improve the performance of function f_T from $\mathcal{T}_T$ using knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, when $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

In [18] transfer learning is divided into four scenarios according to how source and target conditions vary:

1. $\mathcal{X}_S \neq \mathcal{X}_T$, feature spaces of domains are different, e.g. in one domain picture is represented in pixels and in the other domain is described by text.
2. $P(X_S) \neq P(X_T)$, probability distribution of source and target domains are different, e.g. two spectral archive observe different kinds of stars.
3. $\mathcal{Y}_S \neq \mathcal{Y}_T$, labels of tasks are not the same, e.g. spectra should be assign different labels in target task. This usually occurs with the fourth scenario.

4. $P(Y_S \mid X_S) \neq P(Y_T \mid X_T)$, probability distribution of tasks are different.

4.1 Domain Adaptation

In transfer learning the case when probability distribution of source and target domains are different, $P(X_S) \neq P(X_T)$, is generally known as **domain adaptation**. [18]

Thus domain adaptation considers the setting in which the training and target data are sampled from different distributions. The learning problem consists of finding a function realizing a good transfer of knowledge from $\mathcal{D}_S$ to $\mathcal{D}_T$. That means the model trained on data drawn from distribution $P(X_S)$ generalizes well on data drawn from $P(X_T)$. [19]

Chapter 5 Deep Learning Frameworks

There are tens of deep learning frameworks. This chapter makes overview of such frameworks in order to choose the most suitable for purposes of this thesis. Several aspects have been taken into account:

- extensive documentation
- open source project
- long term support
- scalable and GPU ready
- allows fast experiments
- Python API

The three first points should assure that this work will be maintainable in future. This work is faced with vast amount of data so it is desirable to choose scalable and GPU ready framework. On contrary, one goal is to find great deep neural network architecture which means that fast experimentation is crucial. Last two points support the goal of this thesis to find the best deep neural network architecture and because the author of this work is comfortable with Python.

NVIDIA's website[1] lists popular deep learning frameworks. These include Caffe, TensorFlow, Theano, Torch and Keras. Torch is excluded because it has not Python bindings [20].

5.1 Caffe

Caffe is BSD-licensed C++ deep learning framework with Python and MATLAB bindings. It serves for training and deploying convolutional neural networks and other deep models. By separating model representation from actual implementation, Caffe allows seamless switching among CPU, GPU and other architectures. The project is maintained by Berkeley Vision and Learning Center with help of community of contributors on GitHub[2]. Documentation and examples are available at Caffe website[3]. [21]

According to experimental results in [22] Caffe has one of the best performances on both GPU and CPU platforms.

5.2 TensorFlow

TensorFlow is library developed by Google for expressing machine learning algorithms and an implementation for executing such algorithms. It is licensed under Apache 2.0 open source license and its code is available on GitHub[4]. A computation in TensorFlow

[1] `https://developer.nvidia.com/deep-learning-frameworks`
[2] `https://github.com/BVLC/caffe/`
[3] `http://caffe.berkeleyvision.org/`
[4] `https://github.com/tensorflow/tensorflow`

is described by a directed graph which can be executed with little or no change on CPUs or GPUs. Currently implemented front-ends are C++ and Python. This project has extensive documentation with examples and tutorials[1]. [23]

In order to help development and debugging TensorFlow provides visualization tool called TensorBoard. This tool can visualize computation graphs and summary statistics such as values of loss function, weights and so on.

5.3 Theano

Theano is a Python framework for working with mathematical expression. Even though Theano is not developed as deep learning framework it can be consider as it is because primarily usage is to implement mathematical models in a symbolic way and then used them in machine or deep learning. Theano is open source software and is licensed under BSD license. Its source code is versioned in repository on GitHub[2]. Main features are in language to represent mathematical expressions, a compiler to create code to compute these expressions and library to evaluate them. This structure allows to optimize code for both CPUs and GPUs. [24]

As stated in [24] although Theano is developed and mainly used for research in deep learning it is not a deep learning framework in itself. Several software packages have been developed to provide higher-level user interface for instance Keras, Blocks and Lasagne.

5.4 Keras

Keras is high-level neural networks API. It is written in Python and it runs on top of either TensorFlow or Theano. It focuses on fast prototyping and experimentation. Because it is interface to TensorFlow or Theano it can use both CPU and GPUs. Its website[3] provides good documentation and its code is open source under MIT license on GitHub[4]. [25]

Recently Keras API is integrated into `tf.contrib.keras` module of TensorFlow 1.1.0 and then will be moved into `tf.keras` in version 1.2.0. Therefore, there will be two separate implementations of the Keras specification one written in pure TensorFlow and second compatible with both Theano and TensorFlow. [26]

5.5 Consideration

TensorFlow in version 1.1.0 is chosen for implementation in this thesis. It is the most suitable framework because it has both low level interface and high level Keras interface. Moreover, TensorBoard is great tool for graph and dataset interactive visualizations as it contains built-in PCA and t-SNE dimensionality reduction algorithms. Stated in [27] TensorFlow is currently the most popular and future promising deep learning framework in machine learning community.

[1] `https://www.tensorflow.org/`
[2] `https://github.com/Theano/Theano/`
[3] `https://keras.io/`
[4] `https://github.com/fchollet/keras`

Chapter 6
Classifying Ondřejov Archive

This chapter discusses collection and classification of spectral dataset from Ondřejov CCD700 archive. Overview of a tool used for classification is made and then it shows dataset's properties and statistics.

6.1 Spectral View

Spectral View is web browser tool for classification of Ondřejov CCD700 spectral archive. It was developed to support creation of dataset for this work. It is written in Python using Tornado Web Framework[1] and Motor[2] which is asynchronous driver for MongoDB. Visualizations are created with D3.js[3]. Source code is available on GitHub[4] and documentation is available on Read the Docs[5].

Spectra are divided into emission, absorption, double-peak and unknown classes according to shape of spectral line at Hα (see figure 6.1).

Data import is done through a SSAP Service[6]. These spectra are then split into corresponding classes by a user who is presented with visualization of full spectrum, zoomed Hα spectral line and Gaussian convolved Hα. Classified spectra can be viewed and move between classes on demand.

6.2 Ondřejov Dataset

The resulting dataset consists of 13 335 spectra. Structure of the dataset is shown in table 6.1. Emission and absorption spectra are the most significant part of this dataset. There are 45.77% spectra with absorption line in Hα and 39.75% with emission line. But there are also 11.50% spectra with double-peak in Hα plus 2.98% spectra which cannot be easily classified.

class	count
emission	5 301
absorption	6 103
unknown	398
double-peak	1 533

Table 6.1. Number of spectra in Ondřejov dataset's classes.

All spectra in this dataset contain Hα but they have different wavelength starts and ends. Infimum from all wavelength starts is 6 518.43 Å and supremum from all wavelength ends is 6 732.74 Å.

[1] http://www.tornadoweb.org/en/stable/
[2] https://motor.readthedocs.io/en/stable/
[3] https://d3js.org/
[4] https://github.com/podondra/spectralview
[5] http://spectralview.readthedocs.io/
[6] http://www.ivoa.net/documents/SSA/

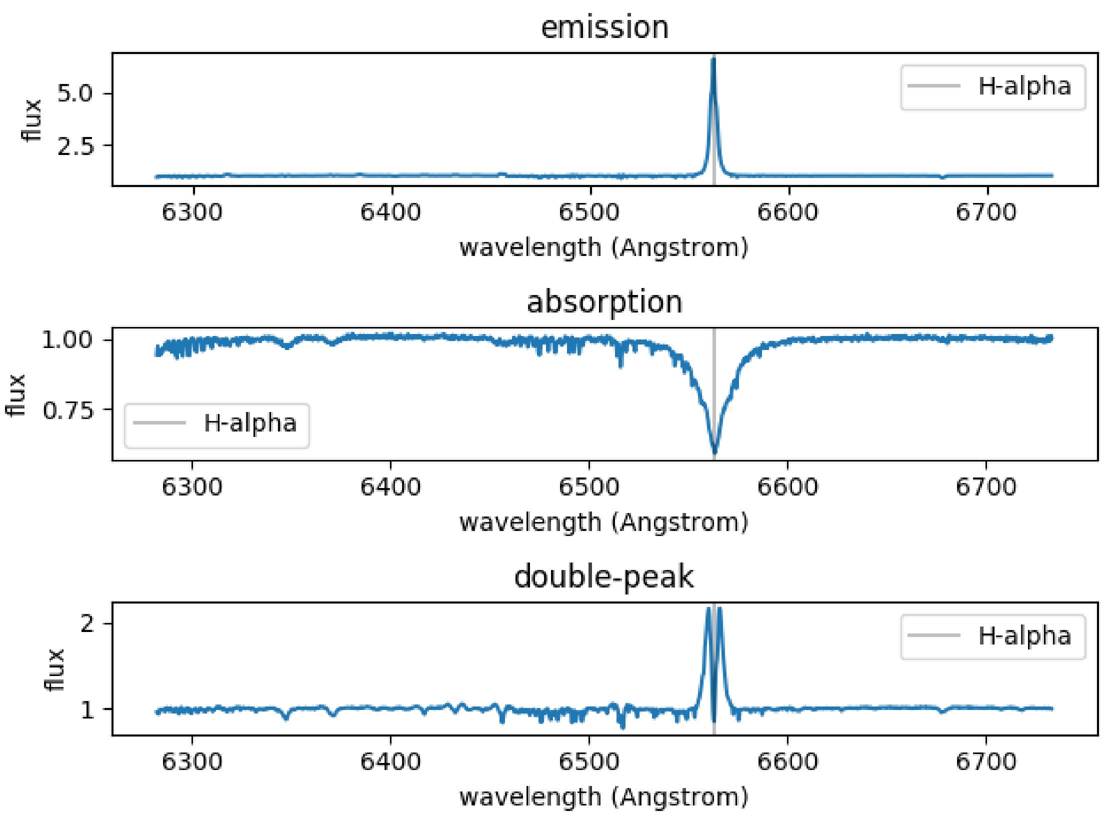

Figure 6.1. Examples from Ondřejov dataset of spectra from emission, absorption and double-peak classes.

Chapter 7
Preprocessing

This chapter describes all preprocessing methods applied on Ondřejov dataset before processing it through a neural network. First section introduces methods to make spectra from Ondřejov similar to spectra from LAMOST. In second section spectra are transformed into form of two dimensional matrix. Third section explains why is scaling of fluxes important.

7.1 Knowledge Transfer

Ondřejov and LAMOST spectrographs have different parameters (as described in sections 2.7 and 2.8). Thus, spectra from the two archives are different (see figure 2.4). Concretely, spectra from Ondřejov archive has much more details than spectra from LAMOST spectrograph and spectra in Ondřejov archive are stored in air wavelengths but LAMOST stores spectra in vacuum wavelengths.

This thesis is concerned with classifying LAMOST data using data from Ondřejov. It aims to extract as much knowledge as possible from Ondřejov data and apply it to classification of LAMOST archive. Transfer learning and domain adaptation (described in chapter 4) are study areas of machine learning that aim to offer methods for dealing with these problems.

The following sections introduce methods that were applied to Ondřejov data. These methods make the two domains more similar and so they deal with differences between the archives.

7.1.1 Wavelength Conversion

Ondřejov and LAMOST archives store wavelengths in air and vacuum wavelength respectively. When spectra of same object from the two archives are plotted on each other they are a bit shifted (see top plot in figure 7.1). Therefore, Ondřejov spectra are converted to vacuum wavelengths according to formulas provided in [28]:

$$\lambda_v = n\lambda_a \tag{7.1}$$

$$\text{where} \quad n = 1 + 8.34254 \cdot 10^{-5} + \frac{2.406147 \cdot 10^{-2}}{(130 - s^2)} + \frac{1.5998 \cdot 10^{-4}}{(38.9 - s^2)} \tag{7.2}$$

$$\text{and} \quad s = \frac{10^4}{\lambda_a} \tag{7.3}$$

where λ_v is vacuum wavelength and λ_a is corresponding air wavelength.

The resulting Ondřejov spectrum after air to vacuum conversion is plotted in bottom plot of figure 7.1.

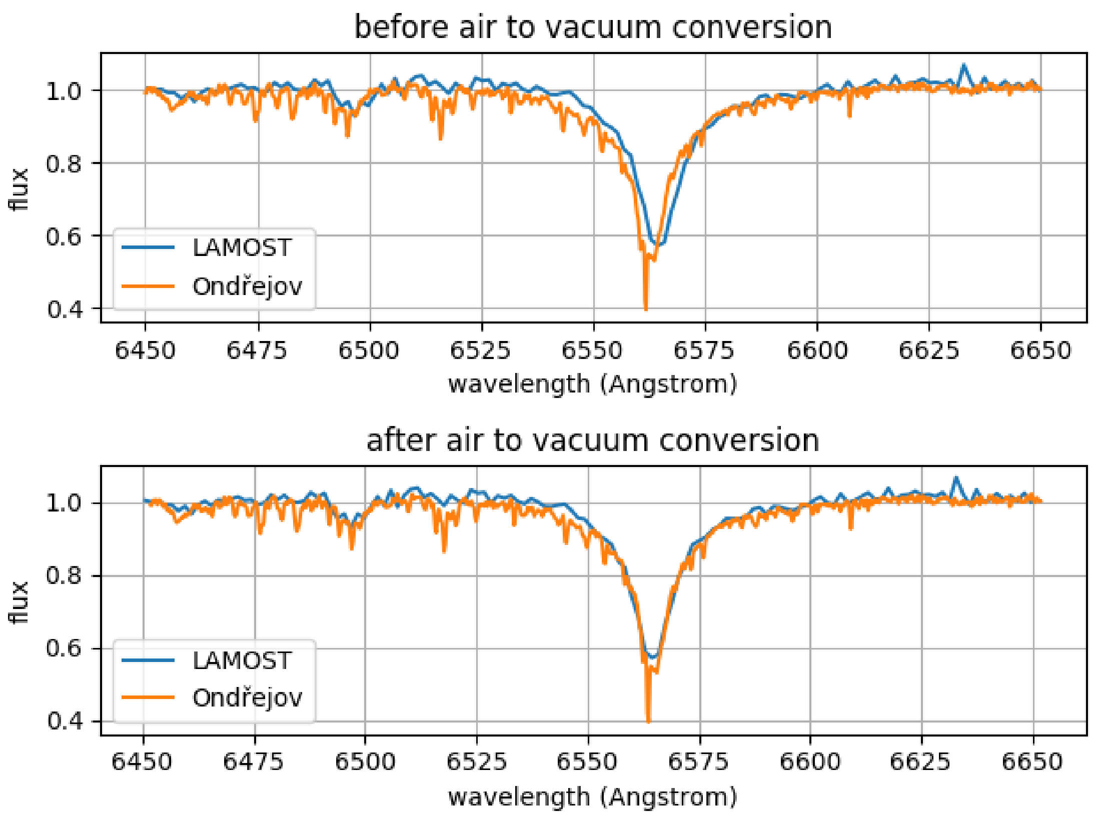

Figure 7.1. Spectrum of object HIP47636 from Ondřejov archive before and after conversion from air to vacuum wavelengths in comparison to spectrum of the same object from LAMOST archive.

7.1.2 Gaussian Blur

According to [29] Gaussian filter smooths away high-frequency detail of images. Spectrum can be seen as one dimensional image. Correctly parameterized Gaussian blur applied to Ondřejov spectrum would reduce the amount of detail and thus make it more similar to spectra from LAMOST archive.

The equation of a one dimensional Gaussian function:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{7.4}$$

where σ is standard deviation in pixels. Kernel generated by this function is convolved with a spectrum resulting in Gaussian blur.

This work uses convolution implementation from Python package **astropy** [30] in version 1.3.1. Standard deviation of value 7 was chosen after visualizing results of convolutions (see figure 7.2). Following code implements functionality described above:

```
from astropy.convolution import Gaussian1DKernel, convolve

gauss_kernel = Gaussian1DKernel(stddev=7)
smoothed_fluxes = convolve(fluxes, gauss_kernel)
```

where `fluxes` is array of spectrum's fluxes. Convolved spectrum's fluxes are stored to `smoothed_fluxes`.

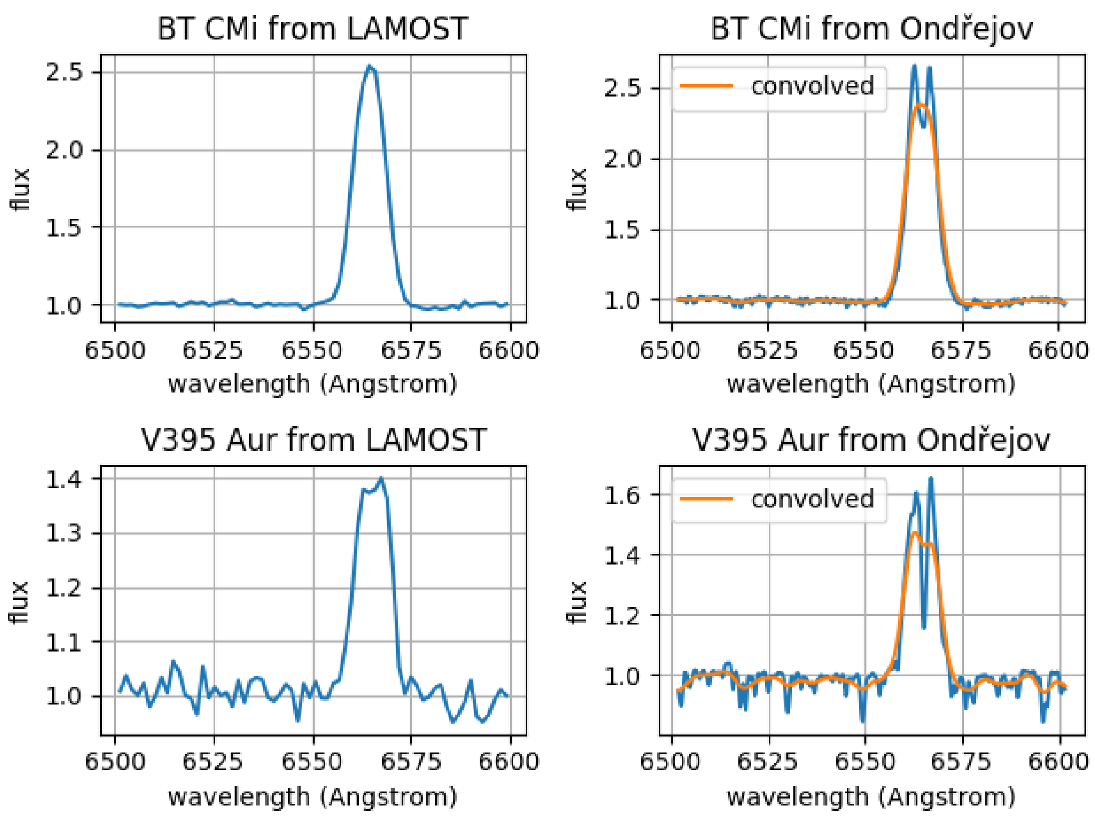

Figure 7.2. Comparison of original BT CMi and V395 Aur spectra from LAMOST and Ondřejov and spectra from Ondřejov convolved by Gaussian kernel with standard deviation of value 7.

7.2 Regridding

To train, validate and test deep neural network two dimensional dataset matrices needs to be created. In a matrix each row is a spectrum sample and each column represents flux in certain wavelength. That means that grid of all spectra need to be changed to same wavelengths.

The wavelengths range is given by Ondřejov dataset from 6 519 to 6 732 (minimum is rounded up and maximum is rounded down to nearest integer value). Ondřejov spectra in this range have 829, 830, 831, or 922, 923 measured fluxes according to this thesis's experiments. This variance is caused by change of spectrograph in Ondřejov observatory (see section 2.7) and the small deviation by the cut of spectrum into the range. LAMOST spectra have 140 measurements in this range inferred from 22 cross-matched spectra.

Because Ondřejov spectra are moved to LAMOST resolution, after air to vacuum conversion and convolution, spectra are regrided to 140 uniform points in range 6519 to 6732. To do regridding linear interpolation is used to infer new fluxes. Here is **NumPy** 1.12.1 [31] code carrying out this operation:

```
import numpy as np

new_wavelengths = np.linspace(6519, 6732, num=140)
new_fluxes = np.interp(new_wavelengths, old_wavelengths, old_fluxes)
```

where `old_wavelengths` variable holds old wavelengths and `old_fluxes` variable holds corresponding fluxes of a spectrum.

7.3 Spectra Scaling

Spectra in Ondřejov dataset were classified according to their shape. But the spectra fluxes have different intensities. Especially, emission spectra tend to have high fluxes. It causes that spectra do not map to same space after dimensionality reduction (see scatter plots 8.1). Therefore, all fluxes of each spectrum are scaled.

Two popular forms of scaling are bound fluxes into range (min-max scaling) and standardization of spectrum to have zero mean and unit variance. As show in histograms 7.3 originally the fluxes are in range $(-0.11, 378.23)$, min-max scaling would bound them to chosen range and after standardization they are in range $(-8.15, 8.25)$.

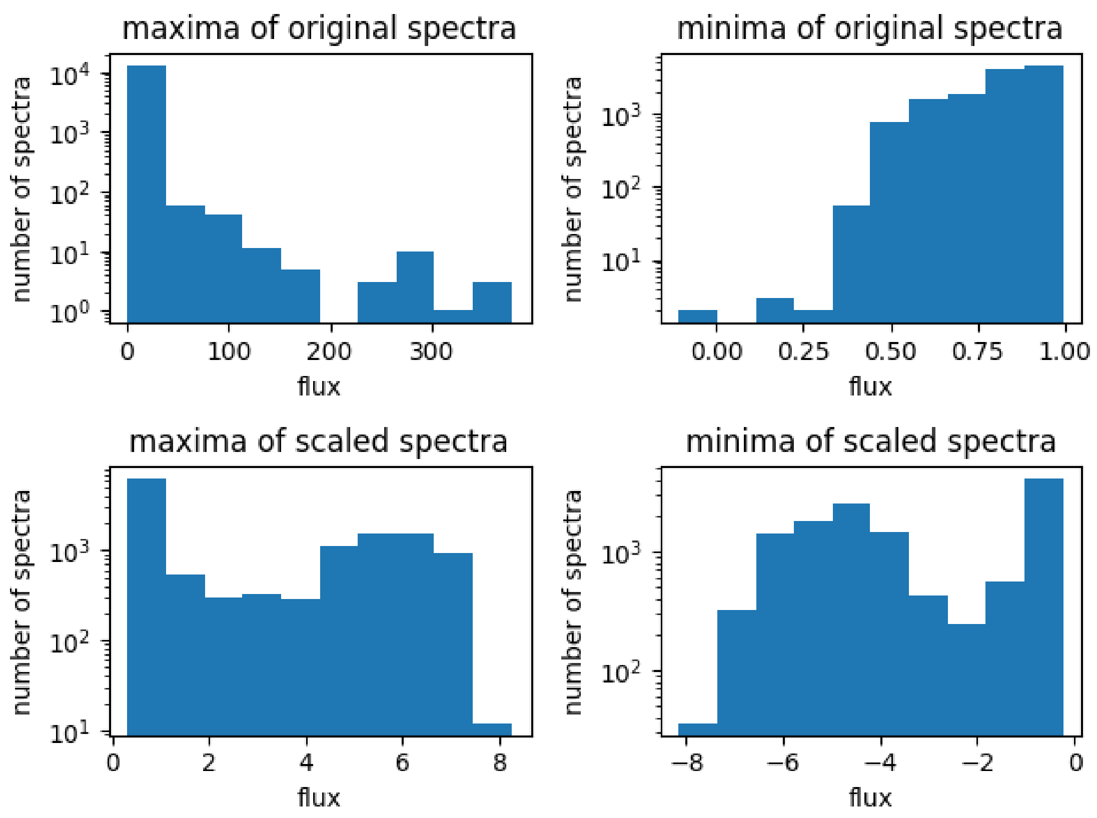

Figure 7.3. Histograms of maxima and minima spectra fluxes before and after standardization to zero mean and unit variance.

The two methods seem to work similar. In this work scaling to zero mean and unit variance is used. Scaling of spectra fluxes is done with **Scikit-learn** 0.18.1 [32]:

```
import sklearn.preprocessing

X_scaled = sklearn.preprocessing.scale(X_original, axis=1)
```

where `X_original` is input matrix of size $n \times 140$ (n is number of spectra samples) containing original dataset. Each row of the matrix is the 140 fluxes of a spectrum.

Chapter 8
Dimensionality Reduction

After the preprocessing methods mentioned in chapter 7 each spectrum is a point in **140-dimensional space**. To better understand the data PCA and t-SNE dimensionality reduction algorithms are applied to reduce each point to 2-dimensional space which can be visualized as scatter plot. Visualization of both Ondřejov and LAMOST data are presented.

8.1 PCA

Principal Component Analysis is statistical technique to transform a number of possibly correlated variables into a smaller number of variables called principal components. [33]

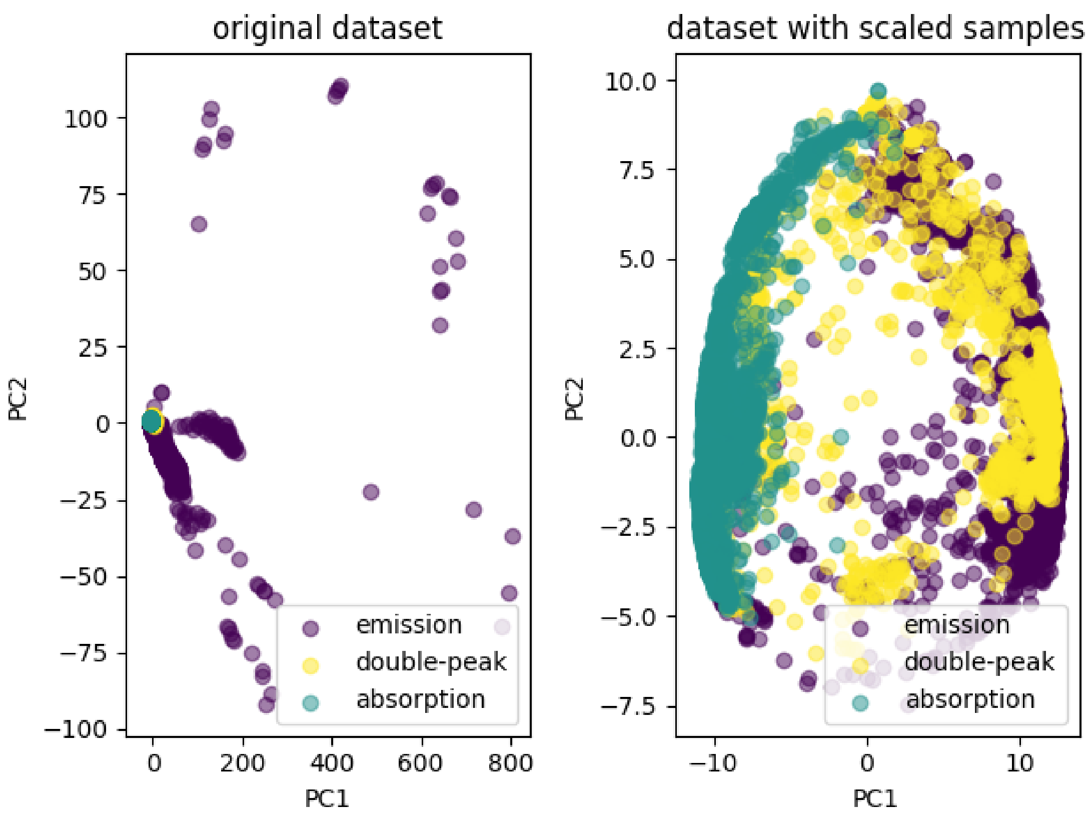

Figure 8.1. On the left size is scatter plot with first two principal components of the original Ondřejov dataset. The plot more that difference in shape show difference in fluxes values. Right plot displays result of PCA applied to scaled variant of the dataset. After such preprocessing method absorption spectra are oriented in left part of the plot, emission spectra are on the other side and double-peak spectra are spread across the whole plot.

Left plot in figure 8.1 displays the first and the second principal component of Ondřejov dataset which is not standardized. After further analysis the plot express more the intensities in spectra fluxes than the shape. The points far from the dense area in middle-left have high fluxes. But in this work the interesting feature is shape not the difference between in fluxes.

The plot on right side in figure 8.1 shows Ondřejov dataset with suppressed fluxes. The scatter plot then expresses the structure of the dataset clearly. Absorption spectra are on the left side while emission spectra are mostly on the other side. Double-peak spectra are spread across the whole plot. This may implies that the emission and absorption spectra can be separable by linear classifier. But the double-peak spectra are mixed up with the other classes so hopefully a deep neural network can find a representation in which all classes are separable.

Figure 8.2 shows PCA applied to sample of LAMOST spectra plotted with all spectra from Ondřejov. The plot uncovers that most of spectra in LAMOST have absorption lines. A lot of points are not similar to any Ondřejov data. But there are probably also emission and double-peak spectra. This is expected behavior because Ondřejov archive is focused on observing emission and double-peak spectra (see section 6) while LAMOST observation is not specialized.

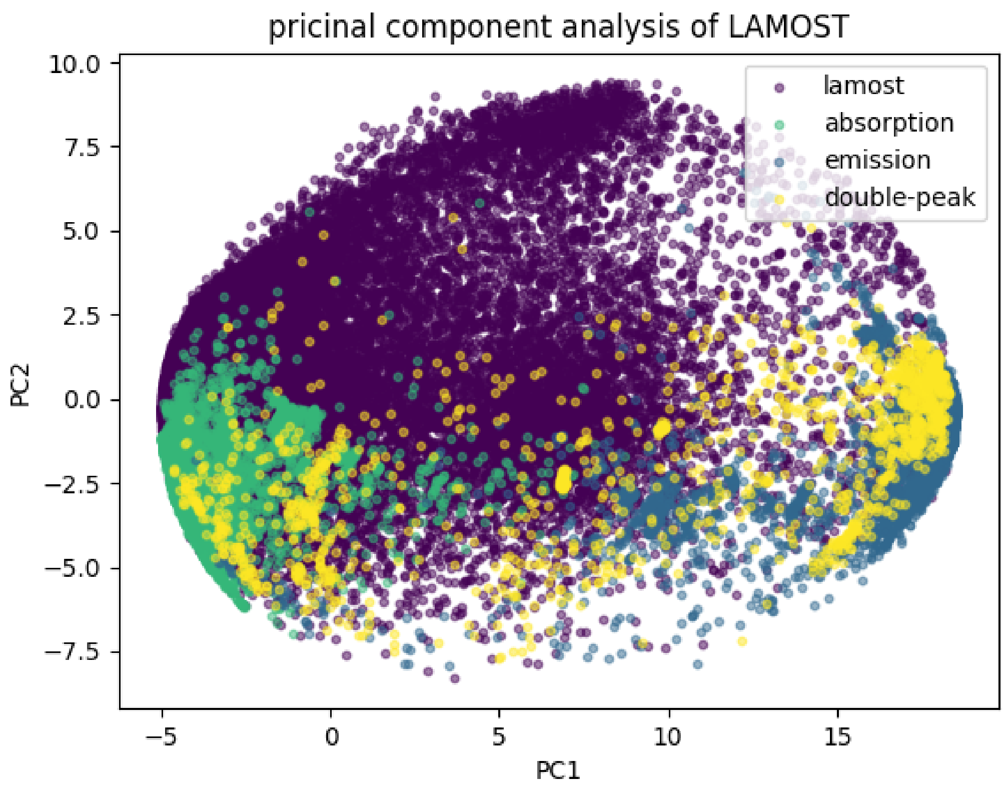

Figure 8.2. The first two principal components of 50 000 random spectra from LAMOST data release 1 and whole Ondřejov dataset. Majority of LAMOST spectra seems to be absorptions but there are some points near to double-peak and emission spectra from Ondřejov.

8.2 t-SNE

T-Distributed Stochastic Neighbor Embedding visualizes high-dimensional data by giving each point a location in a two or three dimensional map. It is kind of Stochastic Neighbor Embedding. Unlike PCA (which is linear technique that focus on keeping the low-dimensional representations of dissimilar points far apart) t-SNE is capable of capturing both global and local structure. [34]

Visualization of Ondřejov dataset with t-SNE is in figure 8.3. Same as experiments in [34], dataset is firstly reduced to 30 dimensions with PCA and then reduce to two dimensions using Scikit-learn implementation of t-SNE. Perplexity, which is t-SNE parameter, is set to value 40.

Figure 8.3 of t-SNE visualization is very similar to PCA scatter plot 8.1. Absorption spectra are on the left side and emission spectra are on right side. Double-peak spectra are more oriented in the middle but they do not form their own cluster.

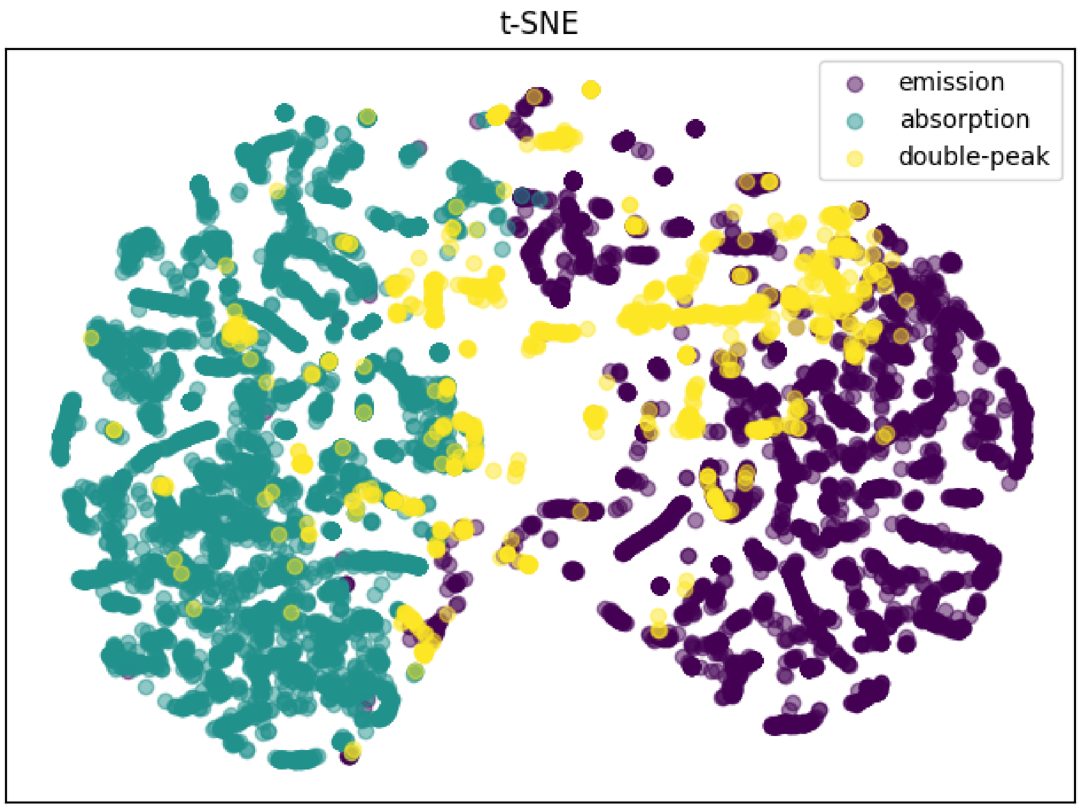

Figure 8.3. Visualization of Ondřejov dataset with t-SNE. The input data are standardized (see section 7.3) and reduced to 30 dimensions with PCA.

Result of t-SNE application on LAMOST data with same parameterization is in visualization 8.4. There are 5 000 random data points from LAMOST and 1 000 random points from Ondřejov. Data are firstly standardized and reduced to 30 dimensions with PCA.

The scatter plot shows a big cluster with absorption spectra on left and smaller cluster of emission spectra on right. Unlike in figure 8.2 there seems to be no spectra from LAMOST too far away from Ondřejov spectra.

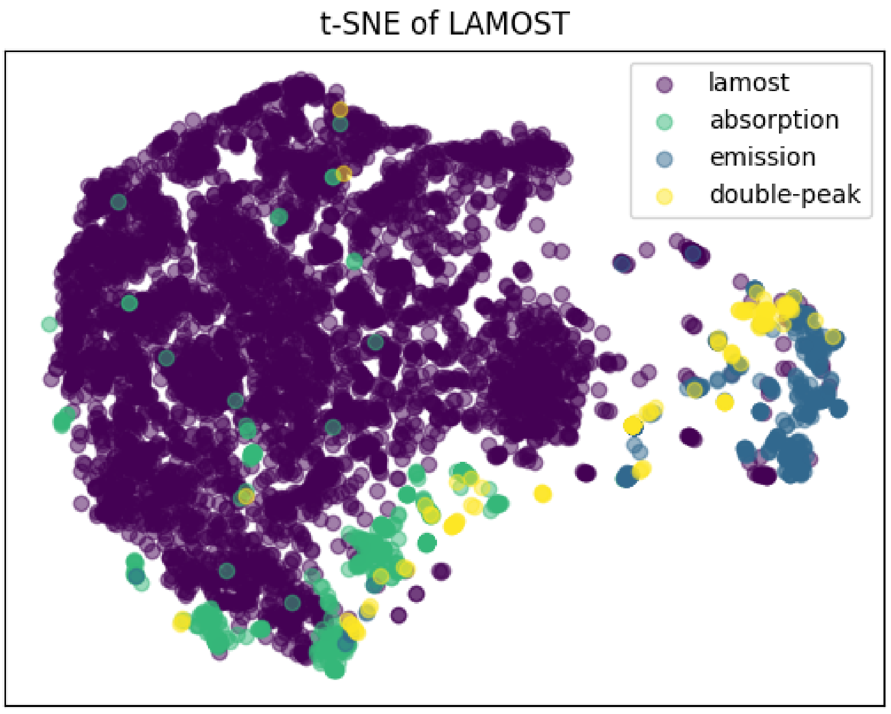

Figure 8.4. Visualization of 5 000 spectra from LAMOST data release 1 and 1 000 spectra from Ondřejov reduced to two dimensions with t-SNE.

Chapter 9
LAMOST Classification

In this chapter all knowledge from previous chapters is gathered and used to classify LAMOST data release 1 which contains spectra from pilot and first year surveys (see table 9). Firstly, architecture and training of a deep neural network is described followed by evaluation and visualizations of results.

9.1 Neural Network Architecture

A feedforward neural network and a convolutional neural network were design. After evaluation on test set the convolutional network was chosen for final classification.

Design choices follow guide provided by [15]. The main idea is to build rather deeper neural network which has a lot of representation power and use dropout (see section 9.1.1) to avoid overfitting. Next sections describe dropout and architectures.

9.1.1 Dropout

Dropout is technique for addressing problem of **overfitting**. The key idea is to randomly drop units from the neural network during training. Dropout created different smaller networks by blocking units of a big network. This prevents units from co-adaption and at test time the neural network with all units but smaller weights is average of all the smaller networks from training. [35]

9.1.2 Feedforward Network

The feedforward network has input layer with 140 units. Then there are 5 hidden layers with 512 units and the last output layer has 3 units with softmax activation function [36] because there are 3 target classes.

Every hidden layer's activation function is ReLU (see section 3.3). During training dropout (see subsection 9.1.1) is applied with a unit dropout probability set to 0.5 as recommended by [35].

9.1.3 Convolutional Network

Architecture of convolutional network for spectra classification in this work is inspired primarily by VGGNet [37], AlexNet [38] and ZFNet [39]. These networks are design to process three dimensional images counting RGB color channels. Spectrum can be understood as one dimensional image. Therefore the architecture needs to be suited to this difference.

Sketch of architecture diagram is in figure 9.1. The input layer has 140 units and output layer has 3 units with softmax activation [36]. In the middle there are convolutional layers all with filter size 3 pixels and no padding. First 2 layers have 64 of the filters. Second 2 layers have 128 of them and the last 2 layers have 256 filters. After each pair of convolutional layers is a max-pooling layers of size 2 pixel with stride 2. The last max-pooling layer is followed are 2 fully-connected layer with 512 units each and dropout with probability 0.5 [35] applied in training. All hidden layers use ReLU activation function.

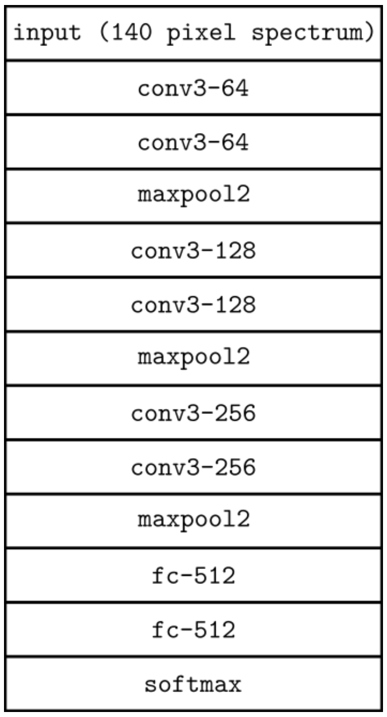

Figure 9.1. Architecture of the convolutional neural network. Convolutional layers are mark as `conv3` where the number 3 means the size of filter in pixels. The mark is followed by dash and a number which specifies count of filters. `maxpool2` are max-pooling layers with pool size 2, stride 2 and no padding. Fully-connected layer with 512 units are `fc-512` and `softmax` is the output layer.

9.2 Training Details

The process of babysitting the neural network training is described in this section. Dataset split and balancing is introduced. Then the hyperparameters of training the network are described. Lastly the networks' performance is evaluated.

9.2.1 Dataset Split

This subsection shows how the Ondřejov dataset is split into training, validation and test set. These sets are required for training, tuning and evaluating any neural network. All split are done as stratified sampling so that the distribution of samples' number in a class is kept across sets.

Test set on which neural networks are evaluated contains 10% of all data. Validation set which serves for hyperparameter and architecture optimization is 20% of remaining data. Training set is composed from the rest of data and its purpose is for training networks' weights. Exact numbers of spectra in each set is in table 9.1.

set	emission	absorption	double-peak	total
train	3 817	4 393	1 104	9 314
validation	954	1 099	276	2 329
test	530	611	153	1 294

Table 9.1. Exact numbers of samples in train, validation and test set divided according to emission, absorption and double-peak classes.

9.2.2 SMOTE Balancing

SMOTE is shortcut for **Synthetic Minority Over-sampling Technique**. It is over-sampling approach in which the minority class is over-sampled by creating **synthetic** samples. A new sample is created along line from a sample to its all or any k nearest neighbors from same class. Difference between feature vectors of sample under consideration and nearest neighbor is taken. It is multiply by random number between 0 and 1. Finally it is added to the sample. [40]

In this thesis SMOTE balancing is used to balance training set because the absorption class which can be considered as normal class has the biggest number of samples while the emission and double-peak classes has significantly less samples. Emission and double-peak spectra which are the abnormal behaviors then tend to be discriminated by the neural network.

Confusion matrix 9.2 is proof of this. The matrix shows 92% accuracy on double-peak spectra of the feedforward neural network on imbalanced dataset. In contrast to 94% accuracy when training on SMOTE balanced dataset (see confusion matrix on the left in figure 9.4).

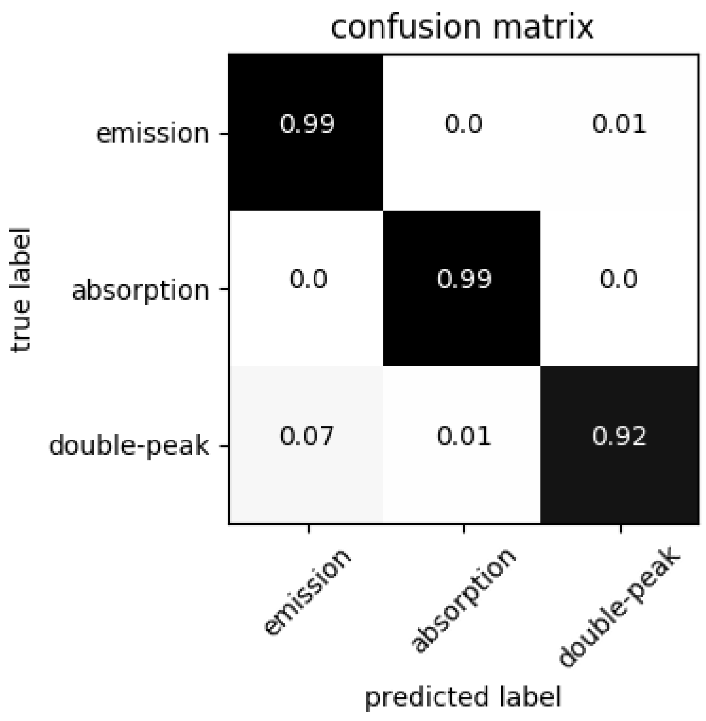

Figure 9.2. Confusion matrix evaluated on validation set of feedforward network trained on imbalanced dataset. Note the low double-peak spectra accuracy.

In this work implementation of SMOTE from Scikit-learn contribution Python package **imbalanced-learn** [41] in version 0.2.1 is used:

```
import imblearn.over_sampling

N_CLASSES = 3
smote = imblearn.over_sampling.SMOTE()
for _ in range(N_CLASSES - 1):
    X, y = smote.fit_sample(X, y)
```

where `X` is input and also output matrix with training data and `y` is vector of corresponding labels.

9.2.3 Training Setting

Training a neural networks is an optimization problem. The function to optimize is **categorical cross-entropy** loss function (see online book [36]).

To optimize to the loss function **Adam** optimizer is used. It is an algorithm for first-order gradient-based optimization of stochastic objective functions. Parameters are set to values recommended by its paper (learning rate $\alpha = 0.001$, decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$, fuzz factor $\epsilon = 10^{-8}$). These are also the defaults in Keras implementation which is available in TensorFlow. [42]

Figure 9.3 shows the CNN's progress of accuracy and loss over 250 epochs (the training set was presented to the network for learning 250 times). The increase of accuracy is fast and training and validation accuracies are close to each other so there is little overfitting.

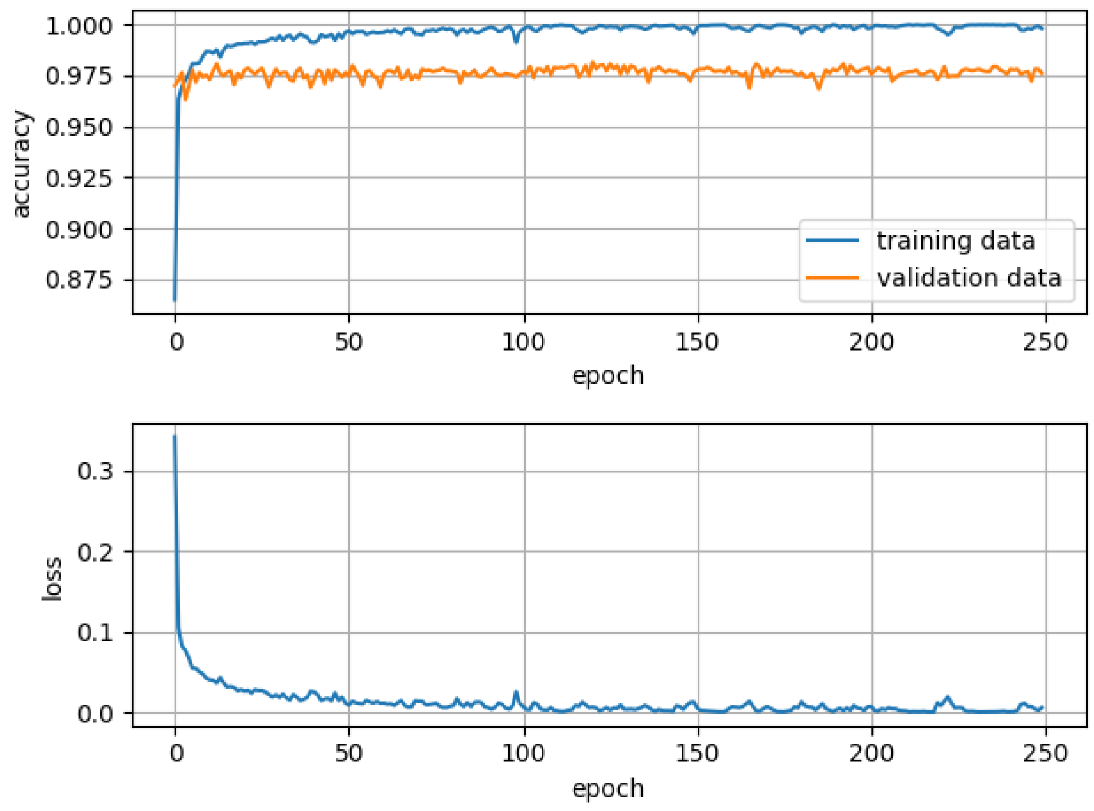

Figure 9.3. Convolutional network training statistics during 250 epochs with batch size 256. The top plot shows training and validation accuracy and the bottom plot shows decrease of loss.

In TensorFlow this implements code:

```
import tensorflow.contrib.keras as keras

model = keras.models.Sequential([
    # definition of neural network
])
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
model.fit(
    X_train, y_train_one_hot, epochs=250, batch_size=256
    validation_data=(X_validation, y_validation_one_hot)
)
```

where `X_train` and `X_validation` are matrices with spectral data, `y_train_one_hot` and `y_validation_one_hot` are classes one hot matrices.

9.2.4 Evaluation

Networks are evaluated on the test set. From confusion matrices in figure 9.4 is clear that convolutional network perform slightly better. CNN predicted correctly 99% of emission spectra while the feedforward network only 98%. Both predicted 99% of absorptions correctly. In double-peak spectra prediction convolutional network has accuracy 95% and feedforward network 94%. Thus convolutional neural network is chosen for classification of spectra from LAMOST.

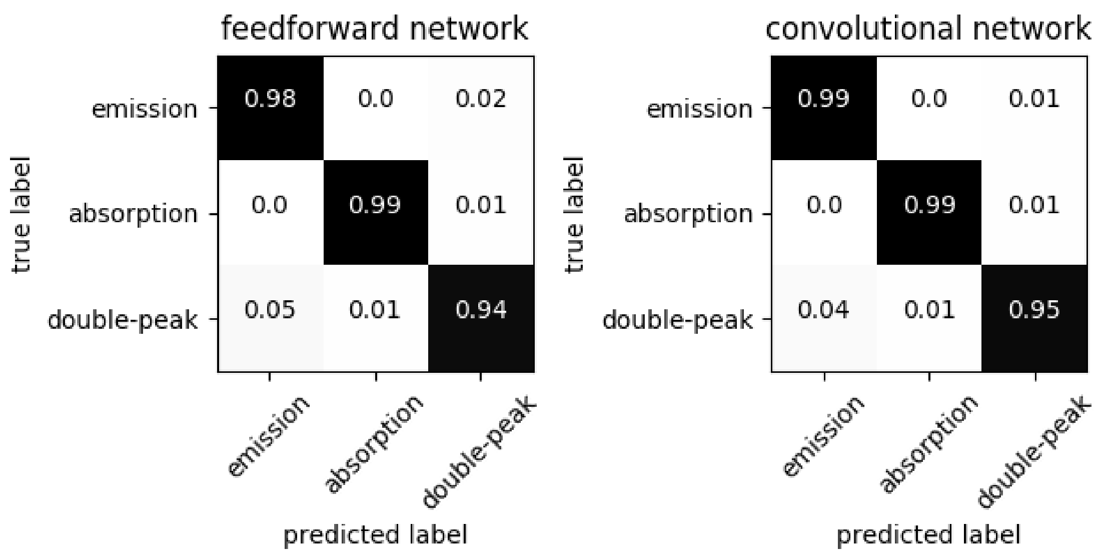

Figure 9.4. Confusion matrices of both networks' predictions on test set. CNN (right matrix) performs slightly better than feedforward network.

9.3 Performance and Scalability

Recent advance in deep neural network research would not be possible without performance improvement of general purpose GPUs. [38] This thesis makes similar observation. The convolutional network was experimentally trained on GPU GeForce GTX 980 and two Intel Xeon CPUs. The GeForce GTX 980[1] has 2 048 CUDA cores and 4 GB of memory. The two Intel Xeon CPUs E5-2403[2] has 8 cores in total and 24 GB of available RAM. Results are presented in table 9.2.

architecture	user time	sys time	wall time
2 Intel Xeon E5-2403	346 min 35 s	41 min 12 s	99 min 36 s
GeForce GTX 980	4 min 1 s	22 s	3 min 39 s

Table 9.2. Comparison of training convolutional neural network on GPU and CPU. The training is done on whole Ondřejov dataset, number of epochs is 100 and batch size is 256. In terms of wall time (elapsed real time) there GPU is roughly 27 times faster.

Table 9.2 confirms the need of GPU computational power for deep neural networks training. Training on GPU is 27 times faster than on CPU. For large networks there might be memory problems on GPU but the net design in this thesis does not require a lot of memory. Moreover TensorFlow support distributed computation so the whole system may scale to more GPU cards on demand. [23]

9.4 Results and Visualizations

For the final classification the convolutional network (diagram 9.1) is trained on whole Ondřejov dataset. The training was stopped after **229 epochs** with **99.96%** accuracy on training set and lasted only **8 minutes 21 seconds**.

Prediction of **2 202 000 spectra** from LAMOST data release 1 has taken **1 minute 36 seconds**. The spectra are interpolated to same grid as spectra Ondřejov dataset (see section 7.2) and standardized (see section 7.3). The number of spectra classified into the classes is in table 9.3.

class	number of spectra
emission	158 115
absorption	1 898 095
double-peak	145 790

Table 9.3. Counts of LAMOST spectra split according to predicted classes.

The table 9.3 shows that the convolutional network reduced the number of possibly interesting object (emission or double-peak) from 2 202 000 to 303 905. That is about 14% from whole LAMOST data release 1. While visualizing spectra according to predicted classes good representatives can be found. Figures 9.5 and 9.6 show examples of correctly classified spectra from the interesting emission and double-peak classes. But

[1] http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-980

[2] http://ark.intel.com/products/64615/Intel-Xeon-Processor-E5-2403-10M-Cache-1_80-GHz-6_40-GTs-Intel-QPI

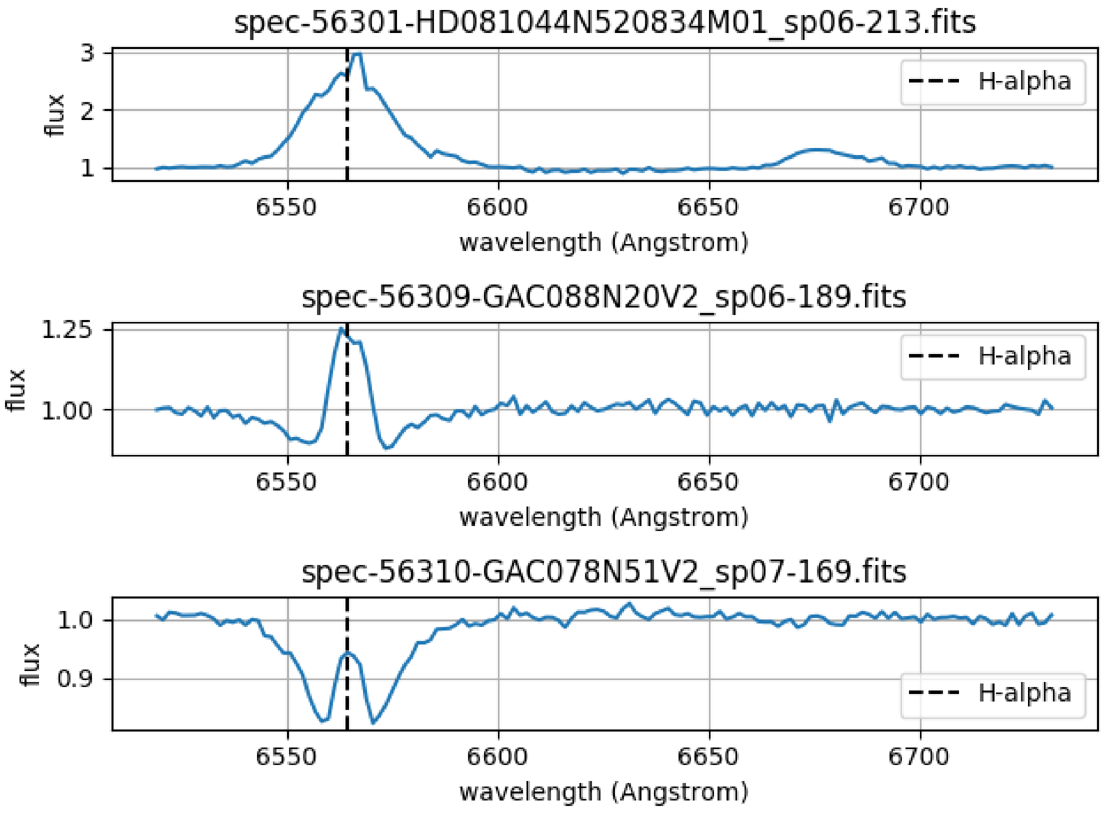

Figure 9.5. Three correctly predicted emission spectra from LAMOST data release 1.

there are also noisy spectra of different kinds mix up with good class candidates. Especially in double-peak class where the noise near to absorption line in often incorrectly consider as double-peak.

This imperfection of the deep convolutional classifier is probably caused by insufficient training set. The Ondřejov dataset and Ondřejov archive in general is composed of not noisy well captured spectra. Therefore the neural network is not trained to recognize noisy or damaged spectra and put them apart. It has to predict them as one of the three classes.

Principal component analysis of LAMOST data space in figure 8.2 also proofs the Ondřejov dataset limitation. Although the points from Ondřejov dataset used during training cover most of the LAMOST data there are some areas where the dataset has no points at all. These are probably the points were the classifier misjudge spectra because it has no prior knowledge about them.

To overcome this problem either a method which can filter the noisy spectra out should be used or the dataset should be extended with new class where should the classifier put noisy spectra.

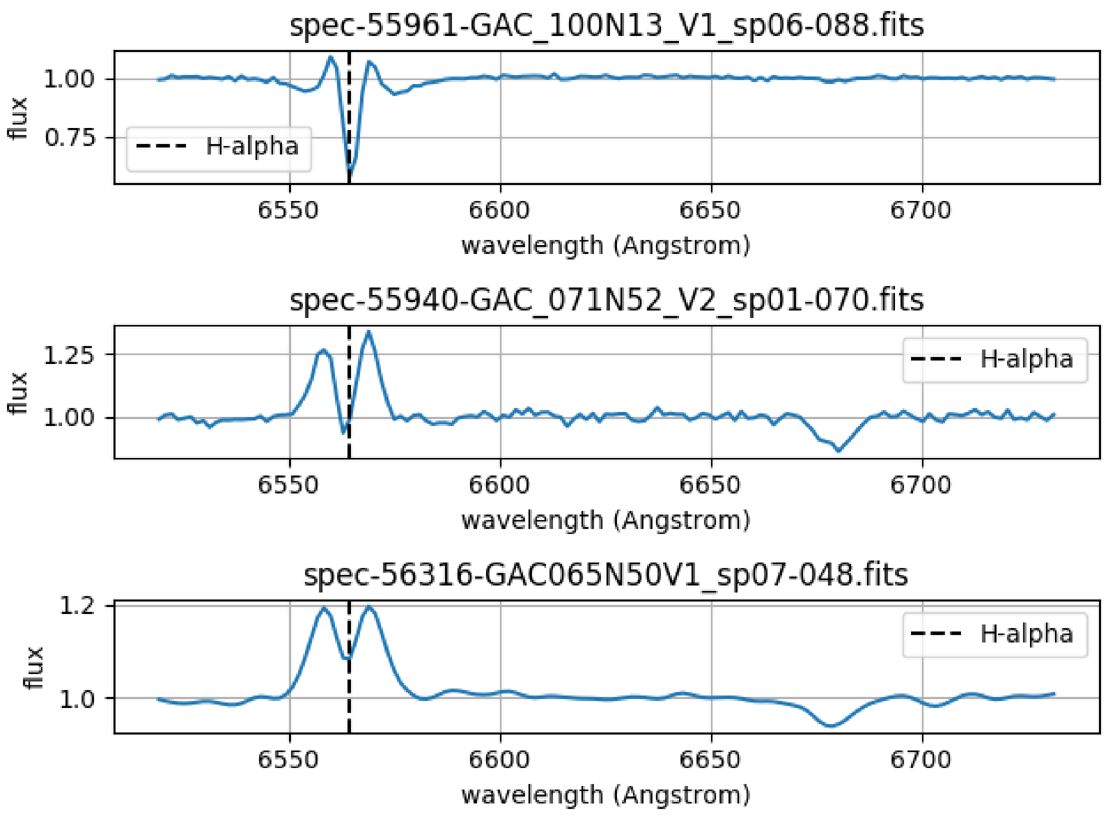

Figure 9.6. Three correctly predicted double-peak spectra from LAMOST data release 1.

Chapter 10
Conclusion

Goal of this work was to classify emission-line spectra in LAMOST using deep neural network trained on spectra from Ondřejov.

For classification three classes of absorption, emission and double-peak spectra were chosen. Training dataset from Ondřejov spectra was created containing 13 335 spectra. Then it was found that to transfer Ondřejov spectra to LAMOST domain wavelength conversion and Gaussian blur with standard deviation of value 7 should be applied to all spectra from Ondřejov. Preprocessing techniques as regridding and spectra scaling were used to make the data suitable for neural networks learning.

PCA and t-SNE dimensionality reduction algorithms have shown that absorption and emission spectra should be easily separable but double-peak spectra require more sophisticated method as deep learning.

Deep convolutional network was evaluated as best choice of architecture for spectra classification. It architecture and training was described. Finally results of classification with the network were presented. The LAMOST data release 1 containing 2 202 000 spectra was reduced to 303 905 candidates (emissions and double-peaks). Although these candidates contains a lot of interesting emission-line spectra deserving further analysis there are plenty of noisy misclassified spectra especially in double-peak predictions.

In future it would be possible to reduce the number of noisy misclassified spectra either by extension of the Ondřejov dataset or by method which could identify noisy spectra.

References

[1] Gregory L. Vogt. *Space-Based Astronomy: An Educator Guide with Activities for Science, Mathematics, and Technology Education* . 2001.

[2] Swinburne University of Technology. *COSMOS - The SAO Encyclopedia of Astronomy*.
`http://astronomy.swin.edu.au/cosmos/`. Accessed: 2017-03-13.

[3] Jeffrey Hall. *The Solar-Stellar Spectrograph*.
`http://www2.lowell.edu/users/jch/sss/article.php?r=t_datared_d_norm` . Accessed: 2017-04-08.

[4] G. M. H. J. Habets, and J. R. W. Heintze. "Empirical bolometric corrections for the main-sequence". *Astronomy and Astrophysics, Supplement*. 1981, 46 193-237.

[5] J. M. Porter, and T. Rivinius. "Classical Be Stars". *Publications of the ASP*. 2003, 115 1153-1170. DOI 10.1086/378307.

[6] *AIASCR VO Services*.
`http://voarchive.asu.cas.cz/`. Accessed: 2017-03-15.

[7] Astronomical Institute AS CR. *Stellar Physics Department*.
`https://stelweb.asu.cas.cz/web/`. Accessed: 2017-03-15.

[8] *LAMOST Survey*.
`http://www.lamost.org/public/?locale=en`. Accessed: 2017-03-15.

[9] *LAMOST Data Release 4* .
`http://dr4.lamost.org/`. Accessed: 2017-03-15.

[10] Dr. Thomas A. McGlynn. *A Primer on the FITS Data Format*.
`https://fits.gsfc.nasa.gov/fits_primer.html`. Accessed: 2017-03-10.

[11] Stuart J. Russell, and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3 edition. Boston: Pearson Education, 2010. ISBN 978-0-13-604259-4.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016.
`http://www.deeplearningbook.org`. Book in preparation for MIT Press.

[13] Arthur Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM J. Res. Dev.*. 1959, 3 (3), 210–229. DOI 10.1147/rd.33.0210.

[14] Tom M. Mitchell. *Machine Learning*. New York City: McGraw-Hill, 1997. ISBN 0-07-042807-7.

[15] Andrej Karpathy. *CS231n*.
`http://cs231n.github.io/`.

[16] Yoshua Bengio LeCun, Yann, and Geoffrey Hinton. Deep learning. *Nature*. 2015, 512 (7553), 436-444.

[17] Sinno Jialin Pan, and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*. 2010, 22 (10), 1345-1359. DOI 10.1109/TKDE.2009.191.

[18] Sebastian Ruder. *Transfer Learning – Machine Learning's Next Frontier*.
`http://sebastianruder.com/transfer-learning/index.html`. Accessed: 2017-04-10.

[19] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. *Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach* . 2011.
`http://www.icml-2011.org/papers/342_icmlpaper.pdf`.

[20] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*. 2015,

[21] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*. 2014,

[22] Shaohuai Shi, Qiang Wang, Pengfei Xu, and Xiaowen Chu. Benchmarking State-of-the-Art Deep Learning Software Tools. *CoRR*. 2016, abs/1608.07249

[23] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan 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, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. 2015.
`http://download.tensorflow.org/paper/whitepaper2015.pdf`.

[24] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*. 2016, abs/1605.02688 19.

[25] Francois Chollet. *Keras Documentation*.
`https://keras.io/`. Accessed: 2017-03-20.

[26] Francois Chollet. *Introducing Keras 2*. 2017.
`https://blog.keras.io/introducing-keras-2.html`. Accessed: 2017-03-20.

[27] Andrej Karpathy. *A Peek at Trends in Machine Learning*.
`https://goo.gl/bDMgY3`. Accessed: 2017-04-10.

[28] Ulrike Heiter. *Air-to-vacuum conversion*.
`http://www.astro.uu.se/valdwiki/Air-to-vacuum%20conversion`. Accessed: 2017-04-25.

[29] Richard Szeliski. *Computer Vision*. London: Springer, 2010. ISBN 978-1-84882-935-0.
`http://szeliski.org/Book/`.

[30] Astropy Collaboration, T. P. Robitaille, E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray, T. Aldcroft, M. Davis, A. Ginsburg, A. M. Price-Whelan, W. E. Kerzendorf, A. Conley, N. Crighton, K. Barbary, D. Muna, H. Ferguson, F. Grollier, M. M. Parikh, P. H. Nair, H. M. Unther, C. Deil, J. Woillez, S. Conseil, R. Kramer, J. E. H. Turner, L. Singer, R. Fox, B. A. Weaver, V. Zabalza, Z. I. Edwards, K. Azalee Bostroem, D. J. Burke, A. R. Casey, S. M. Crawford, N. Dencheva, J. Ely, T. Jenness, K. Labrie, P. L. Lim, F. Pierfederici, A. Pontzen, A. Ptak, B. Refsdal, M. Servillat, and O. Streicher. "Astropy: A community Python package for astronomy". *Astronomy and Astrophysics*. 2013, 558 A33. DOI 10.1051/0004-6361/201322068.

[31] Stéfan vander Walt, S. Chris Colbert, and Gaël Varoquaux. The NumPy array: a structure for efficient numerical computation. *CoRR*. 2011, abs/1102.1523

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, 12 2825–2830.

[33] Mark Richardson. *Principal Component Analysis*. 2009.

[34] Laurens vander Maaten, and Geoffrey E. Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*. 2008, 9 2579-2605.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting . *J. Mach. Learn. Res.*. 2014, 15 (1), 1929–1958.

[36] Michael A Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[37] Karen Simonyan, and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition . *CoRR*. 2014, abs/1409.1556

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. 2012, 1097–1105.

[39] Matthew D. Zeiler, and Rob Fergus. Visualizing and Understanding Convolutional Networks. *CoRR*. 2013, abs/1311.2901

[40] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *CoRR*. 2011, abs/1106.1813

[41] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning . *Journal of Machine Learning Research*. 2017, 18 (17), 1-5.

[42] Diederik P. Kingma, and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*. 2014, abs/1412.6980

Appendix A
Contents of CD

```
/
  README.md                    Markdown file with CD content information
  notebooks/                   directory of Jupyter notebooks and source code
    data/                      data files directory
    spectraldl/                source code of developed Python module
    00-spectroscopy.ipynb      introduction to spectroscopy
    01-data-download.ipynb     download of data from Ondřejov
    02-data-to-hdf5.ipynb      data transformation to HDF5
    03-labeled-data.ipynb      analysis of labeled dataset
    04-cross-matched.ipynb     cross-matching of the two archives
    05-domain-adaptation.ipynb domain adaptation experiments
    06-regridding.ipynb        testing of spectra regridding
    07-dataset.ipynb           train, validation and test set splits
    08-dimensionality.ipynb    archives data visualizations
    09-nns.ipynb               networks design and evaluation
    10-convnet.ipynb           classification of LAMOST spectra
    11-candidates.ipynb        candidates visualizations
    Dockerfile                 assemble commands of image used for notebooks
    requirements.txt           list of dependency packages
  src/                         directory of TEX source code files of the thesis
    img/                       thesis figures directory
    *.tex                      TEX source code files of the thesis
    references.bib             BibTEX source code file of references
  text/                        the thesis text directory
    bt-podsztavek.pdf          Bachelor's thesis in PDF format
```

Appendix B
Glossary

API	■	Application Programming Interface
ASCII	■	American Standart Code for Information Interchange
ASU CAS	■	Astronomical Institute of the Czech Academy of Sciences
CCD	■	Charge-Coupled Device
CNN	■	Convolutional Neural Network
CPU	■	Central Processing Unit
CUDA	■	Compute Unified Device Architecture
FITS	■	Flexible Image Transport System
GPU	■	Graphics Processing Unit
HDF5	■	Hierarchical Data Format 5
HDU	■	Header/Data Unit
LAMOST	■	Large Sky Area Multi-Object Fibre Spectroscopic Telescope
PCA	■	Principal Component Analysis
PDF	■	Portable Document Format
RAM	■	Random-Access Memory
ReLU	■	Rectified Linear Unit
RGB	■	Red Green Blue Color Model
SMOTE	■	Synthetic Minority Over-sampling Technique
SSAP	■	Simple Spectral Access Protocol
t-SNE	■	t-Distributed Stochastic Neighbor Embedding