

Vysoká škola ekonomická v Praze  
Fakulta informatiky a statistiky



# **Variační autoenkodér a úlohy pozorování v latentním prostoru**

## **BAKALÁŘSKÁ PRÁCE**

Studijní program: Aplikovaná informatika

Autor: Tomáš Faltejsek

Vedoucí práce: Ing. Ondřej Vadinský, Ph.D.

Konzultant práce: full consultant's name (incl. degrees)

Praha, květen 2023

## **Poděkování**

Thanks.

## **Abstract**

Jedním z předních rysů lidské inteligence je intuice a schopnost představovat si nové objekty. Variační autoenkodér je inovací na poli pravděpodobnostních modelů, umožňující architekturu modelů schopných syntézy zcela nových dat s využitím pozorování atributů v latentním prostoru. Teoretická charakteristika a možnosti využití variačního autoenkodéru jsou předmětem této bakalářské práce.

## **Keywords**

keyword, important term, another topic, and another one

# Obsah

Úvod	9
<b>1 Východiska variačního autoenkodéru</b>	<b>10</b>
1.1 Umělé neuronové sítě . . . . .	11
1.1.1 Perceptron . . . . .	11
1.1.2 Hebbovské učení . . . . .	11
1.1.3 Universal approximation theorem . . . . .	11
1.1.4 Vícevrstvý Perceptron . . . . .	11
1.1.5 Gradient descent . . . . .	11
1.1.6 Backpropagation . . . . .	11
1.1.7 Strojové učení . . . . .	11
1.1.8 Hluboké učení . . . . .	11
1.1.9 Konvoluční sítě . . . . .	11
1.2 Redukce dimenzionality . . . . .	12
1.2.1 The Curse of Dimensionality . . . . .	12
1.2.2 Analýza hlavních komponent . . . . .	12
1.3 Autoenkodér . . . . .	13
1.3.1 Historický pohled . . . . .	14
1.3.2 Autoenkodér s neúplnou skrytou vrstvou . . . . .	14
1.3.3 Autoenkodér s rozšířenou skrytou vrstvou . . . . .	16
1.3.4 Stacked autoenkodér . . . . .	16
1.3.5 Hlubuký autoenkodér . . . . .	16
1.3.6 Řídký autoenkodér . . . . .	16
1.3.7 Denoising autoenkodér . . . . .	17
1.3.8 Contractive autoenkodér . . . . .	18
1.3.9 Stochastický enkodér dekodér . . . . .	18
1.3.10 Další druhy autoenkoderů . . . . .	18
1.3.11 Taxonomie autoenkodérů . . . . .	18
1.3.12 Využití Autoenkodéru . . . . .	18
1.4 Pravděpodobnostní modely a inference pomocí variačního Bayese . . . . .	19
1.5 Kullback–Lieblerova divergence . . . . .	20
1.6 Modely využívající latentních proměnných // Latent Variable Models . . . . .	21
<b>2 Variační autoenkodér</b>	<b>22</b>
2.1 Evidence Lower Bound . . . . .	22
2.2 Reparametrizační trik . . . . .	22
2.3 Formalizace . . . . .	22
2.4 Model umělé neuronové sítě . . . . .	22
2.5 Nedostatky a omezení . . . . .	22

2.6	Rozšíření a aktuální stav poznání . . . . .	22
2.7	Pozorování v latentním prostoru . . . . .	22
<b>3</b>	<b>Úlohy pozorování v latentním prostoru</b>	<b>23</b>
3.1	Generativní modelování obrazových dat . . . . .	23
3.2	Rekonstrukce obrazových dat . . . . .	23
3.3	Interpolace vět . . . . .	23
3.4	Detekce anomálií . . . . .	23
3.5	Syntéza tabulárních dat . . . . .	23
3.6	Komprese . . . . .	23
<b>4</b>	<b>Experimenty s modelem variačního autoenkodéru</b>	<b>24</b>
4.1	Generativní modelování obrazových dat . . . . .	24
4.1.1	Vymezení problémové oblasti . . . . .	24
4.1.2	Datová sada a předzpracování . . . . .	24
4.1.3	Nastavení experimentu . . . . .	24
4.1.4	Návrh modelu . . . . .	24
4.1.5	Evaluace . . . . .	24
4.1.6	Diskuze . . . . .	24
4.2	Interpolace vět . . . . .	24
	<b>Závěr</b>	<b>25</b>
<b>A</b>	<b>Zdrojové kódy modelů</b>	<b>27</b>

# Seznam obrázků

1.1	Obecná struktura Autoenkodéru. Ze vstupu $x$ je enkodérem vytvořen kód $h$ . . .	13
1.2	Jednotlivé moduly architektury umělé neuronové sítě Autoenkodéru. . . . .	13
1.3	Jednoduchá architektura umělé neuronové sítě Autoenkodéru s neúplnou skrytou vrstvou. Skrytá vrstva představuje <i>bottleneck</i> . . . . .	15
1.4	Deep Autoenkodér. . . . .	16

Note: Add a list of figures if the number of figures in the thesis text exceeds 20. A list of diagrams is applicable only if the author distinguishes between a figure and a diagram. The list of diagrams is included if the number of diagrams exceeds 20. This thesis template does not distinguish between a figure and a diagram.

# Seznam tabulek

Note: Add a list of tables if the number of tables used in the thesis exceeds 20.

# Seznam použitých zkratek

**BCC** Blind Carbon Copy

**CC** Carbon Copy

**CERT** Computer Emergency Response  
Team

**CSS** Cascading Styleheets

**DOI** Digital Object Identifier

**HTML** Hypertext Markup Language

**REST** Representational State Transfer

**SOAP** Simple Object Access Protocol

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

**XML** eXtended Markup Language

Note: Add a list of abbreviations if the number of abbreviations used in the thesis exceeds 20 and the abbreviations used are not common.



# Úvod

Introduction is a compulsory part of the bachelor's / diploma thesis. The introduction is an introduction to the topic. It elaborates the chosen topic, briefly puts it into context (there may also be a description of the motivation to write the work) and answers the question why the topic was chosen. It puts the topic into context and justifies its necessity and the topicality of the solution. It contains an explicit goal of the work. The text of the thesis goal is identical with the text that is given in the bachelor's thesis assignment, ie with the text that is given in the InSIS system and which is also given in the Abstract section.

Part of the introduction is also a brief introduction to the process of processing the work (a separate part of the actual text of the work is devoted to the method of processing). The introduction may also include a description of the motivation to write the work.

The introduction to the diploma thesis must be more elaborate - this is stated in more detail in the Requirements of the diploma thesis within the Intranet for FIS students.

Here are some sample chapters that recommend how a bachelor's / master's thesis should be set. They primarily describe the use of the L<sup>A</sup>T<sub>E</sub>X template, but general advice will also serve users of other systems well.

# **1. Východiska variačního autoenkodéru**

## **1.1 Umělé neuronové sítě**

### **1.1.1 Perceptron**

### **1.1.2 Hebbovské učení**

### **1.1.3 Universal approximation theorem**

### **1.1.4 Vícevrstvý Perceptron**

### **1.1.5 Gradient descent**

### **1.1.6 Backpropagation**

### **1.1.7 Strojové učení**

### **1.1.8 Hluboké učení**

### **1.1.9 Konvoluční sítě**

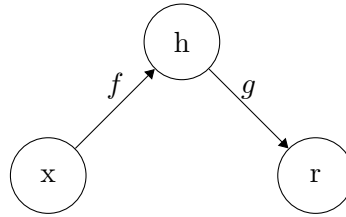
## **1.2 Redukce dimenzionality**

### **1.2.1 The Curse of Dimensionality**

### **1.2.2 Analýza hlavních komponent**

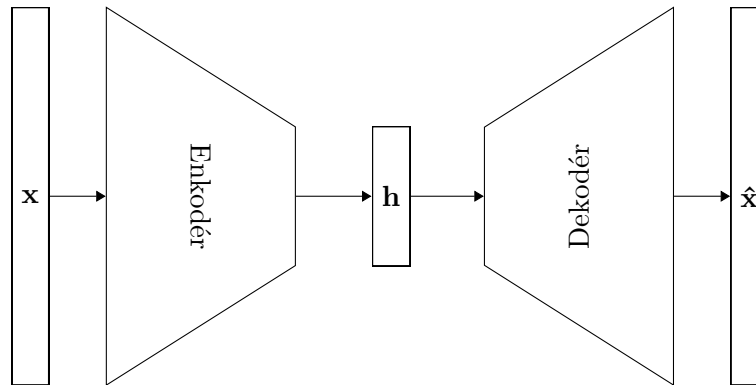
### 1.3 Autoenkodér

Autoenkodér je typ umělé neuronové sítě se schopností učít se efektivní reprezentace vstupních dat bez učitele. Umělá neuronová síť Autoenkodéru má symetrickou strukturu a skrytou vrstvu  $h$ , která popisuje *kód* použitý pro reprezentaci vstupu. Architekturu Autoenkodéru (viz 1.1) lze principiálně rozdělit na dvě části – kódovací funkci  $h = f(x)$ , resp. **enkodér** a dekódovací funkci  $r = g(h)$ , resp. **dekodér**. Hovoříme tedy o typu umělé neuronové sítě s *enkodér-dekodér* moduly. Výstupem enkodéru je **kód** vstupu  $h$ . Výstupem dekodéru je **rekonstrukce** vstupu  $r$ .



Obrázek 1.1: Obecná struktura Autoenkodéru. Ze vstupu  $x$  je enkodérem vytvořen kód  $h$  (funkce  $f$ ). Tento kód je následně dekodérem přetaven na rekonstrukci  $r$  (funkce  $g$ ).

Obecnou strukturu (viz 1.1) lze reprezentovat dopřednou umělou neuronovou sítí. Jejím cílem je **rekonstruovat vstupní data na výstupní vrstvě**. Počet vstupů je tak totožný s počtem neuronů ve výstupní vrstvě umělé neuronové sítě (tedy  $x$  a  $r$  mají stejnou dimenzi).  $h$  může mít *menší* či *větší* dimenzi – volba dimenze  $h$  se odvíjí od požadovaných vlastností Autoenkodéru. Obecná architektura modulů umělé neuronové sítě Autoenkodéru je zachycena v 1.2.



Obrázek 1.2: Jednotlivé moduly architektury umělé neuronové sítě Autoenkodéru.

Autoenkodér je trénován k rekonstrukci jeho vstupů. Pokud by se Autoenkodér naučil jednoduše určit  $x = g(f(x))$  pro každé  $x$ , získali bychom *identitu*, která není patřičně užitečná. Proto je při trénování zavedena řada omezení, jejichž účelem je zabránit možnosti naučení Autoenkodéru perfektně kopírovat vstupní data.

### 1.3.1 Historický pohled

Vícevrstvý Perceptron subsection 1.1.4 je univerzálním aproximátorem subsection 1.1.3 – tedy historicky nalézá uplatnění zejména v klasifikačních úlohách učení s učitelem. Sofistikovaný algoritmus se schopností trénování Vícevrstvého Perceptronu s větším počtem skrytých vrstev stále schází, a to zejména v důsledku problému mizejícího gradientu (*vanishing gradient problem*). Až příchod algoritmu gradientního sestupu subsection 1.1.5, který adresuje problém mizejícího gradientu v aplikacích s použitím konvolučních sítí subsection 1.1.9 a úloh učení se bez učitele, značí počátek moderních metod hlubokého učení. V oblasti hlubokého učení subsection 1.1.8 dochází k emergenci a vývoji řady technik pro řešení úloh učení se bez učitele. V této kapitole je popsána pouze jedna z nich – architektura umělé neuronové sítě založené na *enkodér-dekodér* modulech: Autoenkodér. Autoenkodéry byly poprvé představeny jako způsob pro předtrénování umělých neuronových sítí (formou automatizované extrakce vlastností *feature extraction*). Později Autoenkodéry nalézají uplatnění zejména v úlohách redukce dimenzionality section 1.2 či fúzi vlastností (*feature fusion*).

Nedávné teoretické propojení Autoenkodéru a Modelů využívajících latentní proměnných section 1.6 však vedlo ke vzniku zcela nové architektury neuronové sítě kombinující charakter redukce dimenzionality Autoenkodéru se statistickými metodami odvozování. To vyneslo Autoenkodéry na popředí v oblasti generativního modelování – této architektuře je věnována kapitola chapter 2.

Byť Autoenkodéry vznikly v kontextu hlubokého učení, není pravidlem že všechny modely Autoenkodéru obsahují vícero skrytých vrstev. Následuje rozdělení Autoenkodérů dle struktury umělé neuronové sítě.

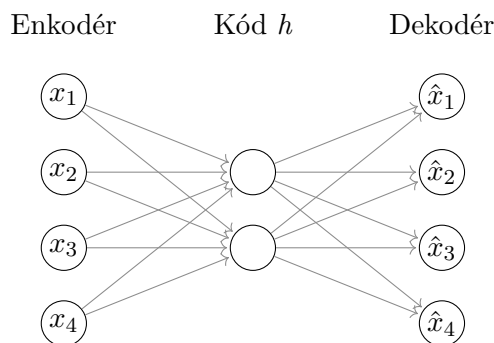
### 1.3.2 Autoenkodér s neúplnou skrytou vrstvou

Autoenkodér s neúplnou skrytou vrstvou (*Undercomplete autoencoder*) je Autoenkodér, jehož dimenze kódu ( $h$ ) je menší, než dimenze vstupu. Tuto skrytou vrstvu  $h$  nazýváme **bottleneck**. Bottleneck je způsob, kterým se Autoenkodér s neúplnou skrytou vrstvou učí kompresované reprezentaci znalostí. V důsledku bottleneck vrstvy je Autoenkodér nucen zachytit pouze ty stěžejní vlastnosti trénovacích dat, které následně budou použity pro rekonstrukci.

Trénovací proces Neúplného autoenkodéru je popsán jako minimalizace ztrátové funkce:

$$L(\mathbf{x}, g(f(\mathbf{x}))), \quad (1.1)$$

kde  $L$  je ztrátová funkce, penalizující  $g(f(\mathbf{x}))$  za *rozdílnost* vůči  $\mathbf{x}$  (např. *střední kvadratická chyba*).



Obrázek 1.3: Jednoduchá architektura umělé neuronové sítě Autoenkodéru s neúplnou skrytou vrstvou. Skrytá vrstva představuje *bottleneck*.

### Od Analýzy hlavních komponent po Autoenkodér

Máme-li lineární dekodér (Autoenkodér používá pouze lineární aktivační funkce) a jako ztrátová funkce  $L$  je použita *střední kvadratická chyba*, pak se Neúplný autoenkodér naučí stejný *vektorový prostor*, který by byl výsledkem Analýzy hlavních komponentů subsection 1.2.2. V tomto speciálním případě lze ukázat, že Autoenkodér trénovaný na úloze kompresované reprezentace znalostí jako vedlejší efekt provedl Analýzu hlavních komponentů.

Důležitým důsledkem tohoto jevu je, že **Autoenkodéry** s nelineární kódovací funkcí  $f$  a nelineární dekódovací funkcí  $g$  **jsou schopny učit se obecnější generalizaci** než u Analýzy hlavních komponent.

Na druhou stranu, má-li Autoenkodér k dispozici příliš mnoho kapacity, může se naučit kopírovat vstupní data na výstupní vrstvu bez extrakce užitečných (charakteristických) vlastností o rozdělení vstupních dat.

### Problém s naučením pouhého identického zobrazení

Extrémním případem je teoretický scénář, ve kterém je Autoenkodér složen z kódu ( $h$ ) o jedné vrstvě a velmi výkonného enkodéru. Takový Autoenkodér by se mohl naučit reprezentovat každý vstup  $x_i$  kódem  $i$ . Dekodér by se pak tyto indexy mohl naučit mapovat zpátky na hodnoty konkrétních trénovacích vzorků dat. Tento příklad se v praxi běžně nenaskytne, nicméně jasně ilustruje, jak může Autoenkodér při úloze kopírování vstupu na výstupní vrstvu selhat naučit se užitečné vlastnosti o vstupních datech, jsou-li restriktce při učení příliš nízké. Proto je třeba Autoenkodéry regularizovat.

Dále tedy budou představeny přístupy k architektuám Autoenkodérů s **využitím regularizace**.

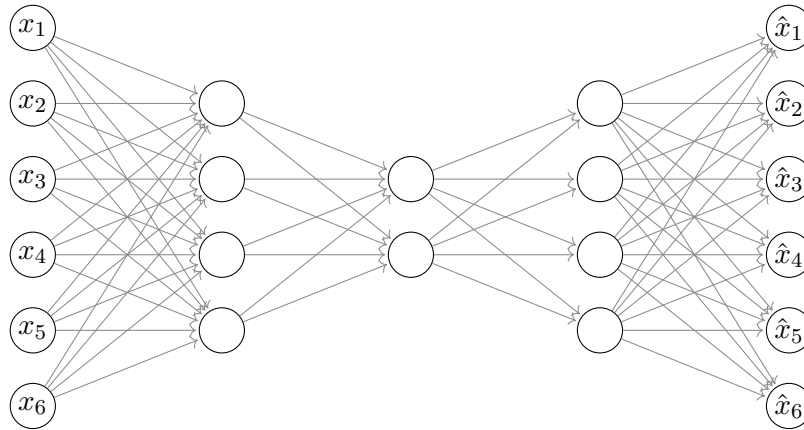
### 1.3.3 Autoenkodér s rozšířenou skrytou vrstvou

Autoenkodér s rozšířenou skrytou vrstvou (*Overcomplete Autoencoder*) je Autoenkodér, jehož počet neuronů ve skryté vrstvě je větší než počet neuronů vstupní (a výstupní) vrstvy.

### 1.3.4 Stacked autoenkodér

### 1.3.5 Hlubuký autoenkodér

Hluboký autoenkodér (*Deep autoenkodér*), je takový Autoenkodér, který má víc než jednu skrytou vrstvu. Hluboký autoenkodér lze trénovat vrstvu po vrstvě formou více *shallow stacked* autoenkodérů zasebou. Nebo jako hlubokou neuronovou síť.



Obrázek 1.4: Deep Autoenkodér.

### 1.3.6 Řídký autoenkodér

Řídká reprezentace dat (*sparsity*) ve strojovém učení znamená, že většina hodnot daného vzorku je nulová. Motivací pro řídkou reprezentaci dat ve strojovém učení je napodobení chování buněk v primární zrakové oblasti (*V1*) mozku savců. Konkrétně schopnosti odhalit a uložit efektivní kódovací strategie pozorovaných vjemů.

Pro sestavení Řídkého autoenkodérů je tedy nutné představit omezení (regularizační prvek) hodnot aktivací neuronů ve skryté (kódovací) vrstvě  $\mathbf{h}$  (resp. počtu aktivních neuronů ve skryté vrstvě).

Řídký autoenkodér (*Sparse Autoencoder*) je Autoenkodér, jehož ztrátová funkce je rozšířena o penalizaci řídkosti kódovací vrstvy  $\mathbf{h}$  (tzv. *sparsity penalty*) vztahem  $\Omega(\mathbf{h})$ :

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}), \quad (1.2)$$



kde  $\Omega$  je **regularizační prvek**, jehož cílem je přiblížit hodnoty aktivací neuronů kódovací vrstvy k cílové hodnotě (a zabránit přeučení). Chceme tak penalizovat neurony kódovací vrstvy, které se aktivují příliš často.

Běžně lze  $\Omega$  stanovit následovně. Mějme Bernoulliho náhodnou proměnou  $i$  modelující aktivace neuronů skryté (kódovací) vrstvy – můžou tedy nastat dva stavy: neuron skryté vrstvy je buď aktivován, nebo není aktivován. Pro konkrétní vstup  $x$  dostaneme:

$$\hat{p}_i = \frac{1}{|S|} \sum_{x \in S} f_i(x), \quad (1.3)$$

kde  $f = (f_1, f_2, \dots, f_c)$ ,  $c$  je počet neuronů skryté (kódovací) vrstvy a  $\hat{p}_i$  je průměrná aktivací hodnota neuronu skryté vrstvy (resp. střední hodnota příslušného Bernoulliho schématu).

Dále mějme  $p$  jako cílové rozdělení aktivací. Kullback-Leibnerova divergence mezi náhodnou proměnnou  $i$  a  $p$  pak udává rozdíl obou rozdělení:

$$KL(p \parallel \hat{p}_i) = p \log \frac{p}{\hat{p}_i} + (1 - p) \log \frac{1 - p}{1 - \hat{p}_i}. \quad (1.4)$$

Výsledný penalizační prvek  $\Omega$  pro Řídký autoenkodér má tedy následující podobu:

$$\Omega_{\text{RAE}}(W, b; S) = \sum_{i=1}^c KL(p \parallel \hat{p}_i), \quad (1.5)$$

kde průměrná hodnota aktivací  $\hat{p}_i$  závisí na parametrech enkodéru a množině trénovacích dat  $S$ .

Přičtením tohoto penalizačního prvku ke ztrátové funkci (a následnou minimalizací celkové ztrátové funkce) je Autoenkodér nucen **omezit počet aktivních neuronů v skryté (kódovací) vrstvě**. V důsledku tohoto omezení pak každý neuron skryté vrstvy reprezentuje nějakou **salientní vlastnost** vstupních dat (a rovněž je zamezeno naučení pouhé identity section 1.3.2).

### 1.3.7 Denoising autoenkodér

Autoenkodéry běžně minimalizují funkci ve tvaru:

$$L(\mathbf{x}, g(f(\mathbf{x}))), \quad (1.6)$$

kde  $L$  je ztrátová funkce penalizující  $g(f(x))$  za odlišnost od  $\mathbf{x}$  (např. Euklidovská norma jejich rozdílů). Jak ale bylo ukázáno v section 1.3.2, to umožňuje  $f \circ g$  naučit se být pouhou identitou.

Z toho důvodu Denoising autoenkodér (*Denoising Autoencoder*) minimalizuje funkci:

$$L(\mathbf{x}, g(f(\tilde{\mathbf{x}}))), \quad (1.7)$$

kde  $\tilde{\mathbf{x}}$  je kopií  $\mathbf{x}$  která byla úmyslně poškozena (např. přidáním šumu, smazáním části vstupních neuronů).

Denoising autoenkodér se tedy musí naučit toto poškození odstranit a rekonstruovat tak původní vstup (namísto pouhého naučení se identitě).

Při trénování Denoising autoenkodéru jsou funkce  $f$  a  $g$  nuceny zachytit implicitní strukturu  $p_{data}(\mathbf{x})$ .

Denoising Autoenkodéry jsou příkladem hned dvou jevů:

- Emergence užitečných vlastností o vstupních datech jako výsledek minimalizace chyby rekonstrukce
- Schopnosti modelů s vysokou kapacitou/rozšířenou skrytou vrstvou fungovat jako Autoenkodér, **za předpokladu že je jim zabráněno naučit se identické zobrazení vstupních dat**

### 1.3.8 Contractive autoenkodér

-> dva podobné vstupy musí mít podobné i kódy!

### 1.3.9 Stochastický enkodér dekodér

### 1.3.10 Další druhy autoenkoderů

### 1.3.11 Taxonomie autoenkodérů

### 1.3.12 Využití Autoenkodéru

- Mapování vysokorozměrných dat do 2D pro vizualizaci
- Učení se abstraktních vlastností o vstupních datech bez učitele, pro následné využití v supervizovaných úlohách
- Komprese

## **1.4 Pravděpodobnostní modely a inference pomocí variačního Bayese**

## 1.5 Kullback–Lieblerova divergence

## **1.6 Modely využívající latentních proměnných // Latent Variable Models**

## **2. Variační autoenkodér**

**2.1 Evidence Lower Bound**

**2.2 Reparametrizační trik**

**2.3 Formalizace**

**2.4 Model umělé neuronové sítě**

**2.5 Nedostatky a omezení**

**2.6 Rozšíření a aktuální stav poznání**

**2.7 Pozorování v latentním prostoru**

## **3. Úlohy pozorování v latentním prostoru**

**3.1 Generativní modelování obrazových dat**

**3.2 Rekonstrukce obrazových dat**

**3.3 Interpolace vět**

**3.4 Detekce anomálií**

**3.5 Syntéza tabulárních dat**

**3.6 Komprese**

## **4. Experimenty s modelem variačního autoenkodéru**

### **4.1 Generativní modelování obrazových dat**

#### **4.1.1 Vymezení problémové oblasti**

#### **4.1.2 Datová sada a předzpracování**

#### **4.1.3 Nastavení experimentu**

#### **4.1.4 Návrh modelu**

#### **4.1.5 Evaluace**

#### **4.1.6 Diskuze**

### **4.2 Interpolace vět**



# Závěr

The conclusion is a mandatory part of the bachelor's / diploma thesis. It contains a summary of the work and comments on the degree of fulfillment of the goal, which was set in the work, or summarizes the answers to the questions that were asked in the introduction.

The conclusion to the diploma thesis must be more elaborate - this is stated in more detail in the Requirements of the diploma thesis within the Intranet for FIS students.

The conclusion is perceived as a chapter, which begins on a separate page and is called the conclusion. The name Conclusion is not numbered. The text of the conclusion itself is divided into paragraphs.

## **Přílohy**

## **A. Zdrojové kódy modelů**