

Univerzita Pardubice  
Fakulta elektrotechniky a informatiky

Detekce a analýza exoplanet s využitím  
distribuovaných výpočtů a umělé inteligence

Michal Struna

Diplomová práce

2021

Univerzita Pardubice  
Fakulta elektrotechniky a informatiky  
Akademický rok: 2020/2021

## ZADÁNÍ DIPLOMOVÉ PRÁCE (projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Bc. Michal Struna**  
Osobní číslo: **I19288**  
Studijní program: **N0613A140007 Informační technologie**  
Studijní obor: **Informační technologie**  
Téma práce: **Detekce a analýza exoplanet s využitím distribuovaných výpočtů a umělé inteligence**  
Zadávací katedra: **Katedra softwarových technologií**

### Zásady pro vypracování

Cílem práce je analýza, návrh a implementace dílčích aplikací tvořících jeden celek, kdy jednotlivé dílčí aplikace jsou na sobě závislé. Výstupní aplikace slouží pro hledání planet mimo sluneční soustavu (tzv. exoplanet) z datasetů, které jsou výstupem z vesmírných teleskopů (např. teleskop Kepler od NASA). Z důvodu výpočetní složitosti a vzhledem k tomu, že datasety obsahují často statisíce či miliony položek, výpočty budou probíhat distribuovanou formou, přičemž předpokladem je, že by uživatelé poskytovali výpočetní výkon svých počítačů pro řešení problémů.

Výstupem praktické části budou 3 dílčí aplikace, tvořící jeden výstupní celek:

- Server s databází (Python, MongoDB) – přiděluje výpočetní úkoly klientům, uchovává persistentní data (objevené planety, uživatele, ...),
- Klientská aplikace (Python) – aplikace, která bude ke stažení a na klientském počítači budou prováděny serverem přidělené výpočty,
- Webová aplikace (TypeScript, React) – tato aplikace bude umožňovat prohlížení položek v databázi, administraci, apod.

Rozsah pracovní zprávy: **50-60 stran**  
Rozsah grafických prací: **-**  
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

- \*MongoDB – <https://docs.mongodb.com/manual/reference/database-references/>
- \*NORDEEN Alex. *MongoDB: Learn in 24 hours*. Guru99, 2020.
- \*Otevřená data NASA – <https://nasa.github.io/data-nasa-gov-frontpage/>
- \*PECINOVSKÝ, Rudolf. *Python: kompletní příručka jazyka pro verzi 3.9*. Praha: Grada Publishing, 2020. Knihovna programátora (Grada). ISBN 978-80-271-1269-2.

Vedoucí diplomové práce: **Ing. Monika Borkovcová, Ph.D.**  
Katedra informačních technologií

Datum zadání diplomové práce: **6. listopadu 2020**  
Termín odevzdání diplomové práce: **15. května 2021**

**Ing. Zdeněk Němec, Ph.D.** v.r.  
děkan

L.S.

**prof. Ing. Antonín Kavička, Ph.D.** v.r.  
vedoucí katedry

V Pardubicích dne 30. listopadu 2020

## Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 9. 5. 2021

Michal Struna

## **Poděkování**

Tímto děkuji vedoucí této práce, Ing. Monice Borkovcové Ph.D., za přínosné rady při vypracovávání, ochotu a snahu dovést tuto práci do úspěšného konce. Velký dík patří i mé rodině, která mě už od dětství podporovala v astronomii a bez které bych se k tomuto tématu nejspíš nedostal.

## **ANOTACE**

Diplomová práce se v teoretické části zabývá jednotlivými metodami detekce a analýzy exoplanet. Dále je obecně popsáno odvětví umělé inteligence a detailněji pak konkrétní algoritmy a postupy z této oblasti, kterými by bylo možné zautomatizovat výše popsané metody. Praktická část se zaměřuje na návrh a implementaci aplikace, jež umožní dobrovolníkům poskytovat výpočetní výkon svých počítačů pro zpracování dat pocházejících z pozorování hvězd tranzitní metodou. Výsledky tohoto zpracování jsou ukládány do persistentního uložení a jsou veřejně přístupné.

## **KLÍČOVÁ SLOVA**

exoplanety, extrasolární planety, kepler, umělá inteligence, python

## **TITLE**

Detection and analysis of exoplanets using distributed computing and artificial intelligence

## **ANNOTATION**

The diploma thesis deals in the theoretical part with individual methods of detection and analysis of exoplanets. Furthermore, the field of artificial intelligence is generally described and in more detail specific algorithms and procedures in this field, which could automate the methods described above. The practical part focuses on the design and implementation of an application that will allow volunteers to provide the computing power of their computers for processing data from star observation using the transit method. The results of this processing are stored in a persistent repository and are publicly accessible.

## **KEYWORDS**

exoplanets, extrasolar planets, kepler, artificial intelligence, python

# OBSAH

Seznam obrázků	10
Seznam tabulek	13
Seznam zdrojových kódů	14
Seznam obrázků	16
Úvod	17
<b>1 Hledání exoplanet</b>	<b>18</b>
1.1 Tranzitní metoda . . . . .	21
1.1.1 Target pixel file . . . . .	22
1.1.2 Světelná křivka hvězdy . . . . .	22
1.1.3 Vyřazení falešně pozitivních výsledků . . . . .	23
1.1.4 Výpočet vlastností planety . . . . .	25
1.1.5 Výpočet hmotnosti planety . . . . .	26
1.2 Metoda radiálních rychlostí . . . . .	27
1.2.1 Výpočet radiální rychlosti hvězdy . . . . .	28
1.2.2 Výpočet hmotnosti planety . . . . .	28
1.3 Astrometrická metoda . . . . .	29
1.4 Přímé zobrazení . . . . .	30
1.5 Gravitační mikročočky . . . . .	30
1.6 Časování pulsarů . . . . .	31
1.7 Časování tranzitů . . . . .	33
1.7.1 Aplikace genetického algoritmu . . . . .	34
<b>2 Umělé neuronové sítě</b>	<b>35</b>
2.1 Neuron . . . . .	35
2.1.1 Vstupy neuronu . . . . .	36
2.1.2 Váhy spojení . . . . .	36
2.1.3 Práh neuronu . . . . .	37

2.1.4	Agregační funkce . . . . .	37
2.1.5	Aktivační funkce . . . . .	37
2.2	Typy neuronových sítí . . . . .	38
2.2.1	Jednoduchý perceptron . . . . .	38
2.2.2	Dopředná vícevrstvá umělá neuronová síť . . . . .	39
2.2.3	Konvoluční neuronová síť . . . . .	40
2.3	Typy vrstev neuronových sítí . . . . .	42
2.3.1	Plně propojená vrstva . . . . .	42
2.3.2	Konvoluční vrstva . . . . .	42
2.3.3	Vrstva Max-Pooling . . . . .	44
2.3.4	Vrstva Flatten . . . . .	45
2.3.5	Vrstva Dropout . . . . .	45
2.4	Učení umělé neuronové sítě . . . . .	46
2.4.1	Algoritmus zpětného šíření chyby . . . . .	46
<b>3</b>	<b>Další oblasti umělé inteligence</b>	<b>48</b>
3.1	Genetický algoritmus . . . . .	49
3.1.1	Inicializace . . . . .	50
3.1.2	Selekce . . . . .	51
3.1.3	Křížení . . . . .	51
3.1.4	Mutace . . . . .	52
<b>4</b>	<b>Použité technologie</b>	<b>53</b>
4.1	TypeScript . . . . .	53
4.1.1	React . . . . .	53
4.1.2	Styled components . . . . .	53
4.2	Python . . . . .	54
4.2.1	Flask . . . . .	54
4.2.2	Astropy . . . . .	54
4.2.3	LightKurve . . . . .	55
4.2.4	Keras . . . . .	55
4.2.5	MongoEngine . . . . .	55
4.3	MongoDB . . . . .	56



4.4	Socket.io . . . . .	56
<b>5</b>	<b>Návrh a vývoj aplikace</b>	<b>57</b>
5.1	Server . . . . .	57
5.1.1	REST API . . . . .	58
5.1.2	Socket.IO API . . . . .	59
5.2	Webová aplikace . . . . .	59
5.3	Klientská aplikace . . . . .	62
5.4	Neuronové sítě . . . . .	64
5.4.1	Klasifikace hvězdy . . . . .	64
5.4.2	Detekce tranzitů exoplanety . . . . .	65
5.5	Databáze . . . . .	65
5.6	Datasety . . . . .	66
5.6.1	Target pixel files . . . . .	66
5.6.2	Hvězdy . . . . .	67
5.6.3	Planety . . . . .	67
5.6.4	Názvy . . . . .	68
5.7	Instalace a spuštění aplikace . . . . .	69
5.7.1	Konfigurace . . . . .	69
5.7.2	Vývoj . . . . .	70
5.7.3	Testování . . . . .	71
<b>6</b>	<b>Rozvržení aplikace</b>	<b>72</b>
6.1	Přehled . . . . .	72
6.2	Databáze . . . . .	73
6.3	Detail systému . . . . .	74
6.4	Objevování exoplanet . . . . .	76
6.5	Autentizace . . . . .	76
	<b>Závěr</b>	<b>78</b>
	<b>Použitá literatura</b>	<b>79</b>
	<b>Seznam příloh</b>	<b>83</b>

# SEZNAM OBRÁZKŮ

Obrázek 1	Četnost objevů exoplanet v jednotlivých letech . . . . .	18
Obrázek 2	Počty objevených exoplanet jednotlivými metodami . . . . .	19
Obrázek 3	Znamé exoplanety dle jejich vlastností a metody objevení . . . . .	19
Obrázek 4	Počty objevených exoplanet dle jejich typu . . . . .	20
Obrázek 5	Přechod planety přes kotouč hvězdy . . . . .	21
Obrázek 6	TPF soustavy Kepler-10 s lineárním a logaritmickým měřítkem . . . . .	22
Obrázek 7	Světelná křivka soustavy Kepler-13 . . . . .	23
Obrázek 8	Dvojhvězda KIC 8262223 . . . . .	24
Obrázek 9	Cefeida KIC 3733346 . . . . .	24
Obrázek 10	Proměnná hvězda KIC 9832227 . . . . .	24
Obrázek 11	Kataklyzmická proměnná hvězda KIC 9406652 . . . . .	24
Obrázek 12	Světelná křivka Kepler-10 . . . . .	24
Obrázek 13	Složená světelná křivka . . . . .	24
Obrázek 14	Globální pohled na tranzit Kepler-10 c . . . . .	25
Obrázek 15	Lokální pohled na tranzit Kepler-10 c . . . . .	25
Obrázek 16	Metoda radiálních vzdáleností . . . . .	27
Obrázek 17	Radiální rychlost hvězdy 51 Pegasi v čase . . . . .	28
Obrázek 18	Obíhání hvězdy a planety kolem společného těžiště . . . . .	29
Obrázek 19	Kolísání hvězdy Gliese 876 s planetou . . . . .	29
Obrázek 20	Hvězda GQ Lupi s přímo pozorovanou exoplanetou . . . . .	30
Obrázek 21	Princip gravitační čočky . . . . .	30
Obrázek 22	Ilustrace gravitační mikročočky OGLE 2003-BLG-235 . . . . .	31
Obrázek 23	Ilustrace pulsaru s obíhající planetou . . . . .	32
Obrázek 24	Nepřavidelnosti v tranzitech planety Kepler-46b . . . . .	33
Obrázek 25	Princip časování tranzitů . . . . .	34
Obrázek 26	Model formálního neuronu . . . . .	36
Obrázek 27	Příklady aktivačních funkcí neuronu . . . . .	38
Obrázek 28	Lineárně separovatelná (vlevo) a neseperovatelná (vpravo) úloha . . . . .	38
Obrázek 29	Zapojení neuronů . . . . .	39
Obrázek 30	Architektura dopředné vícevrstvé umělé neuronové sítě . . . . .	39

Obrázek 31	Ručně psané číslice 9. . . . .	40
Obrázek 32	Vztah mezi počítačovým viděním, strojovým učením a konvoluční sítí	40
Obrázek 33	Příklad konvoluční sítě pro rozpoznávání ručně psaných číslic . . . . .	41
Obrázek 34	Postupné posouvání konvolučního filtru přes vstup . . . . .	42
Obrázek 35	Příklad aplikace konvolučního filtru na vstup . . . . .	43
Obrázek 36	Příklad operace Max-Pooling . . . . .	44
Obrázek 37	Příklad fungování vrstvy Flatten . . . . .	45
Obrázek 38	Příklad fungování vrstvy Dropout . . . . .	45
Obrázek 39	Příklad algoritmu zpětného šíření chyby . . . . .	47
Obrázek 40	Rozdělení umělé inteligence . . . . .	49
Obrázek 41	Někteří z vygenerovaných jedinců GA pro TSP . . . . .	50
Obrázek 42	Příklad křížení jedinců GA pro TSP . . . . .	51
Obrázek 43	Příklad mutace jedince GA pro TSP . . . . .	52
Obrázek 44	Hledání řešení TSP pomocí GA . . . . .	52
Obrázek 45	Komponenty projektu a jejich komunikace . . . . .	57
Obrázek 46	Architektura serveru . . . . .	57
Obrázek 47	Architektura webové aplikace . . . . .	60
Obrázek 48	Princip architektury Redux . . . . .	60
Obrázek 49	Navázání spojení klientské aplikace . . . . .	62
Obrázek 50	Životní cyklus procesu klientské aplikace . . . . .	62
Obrázek 51	Architektura klientské aplikace . . . . .	63
Obrázek 52	Architektura neuronové sítě pro klasifikaci hvězd . . . . .	64
Obrázek 53	Klasifikace hvězdy – trénovací množina a HR diagram . . . . .	64
Obrázek 54	Část trénovací množiny pro detekci tranzitů exoplanet . . . . .	65
Obrázek 55	Část datasetu s vlastnostmi hvězd . . . . .	67
Obrázek 56	Část datasetu s vlastnostmi planet . . . . .	68
Obrázek 57	Stránka Přehled . . . . .	72
Obrázek 58	Statistiky . . . . .	72
Obrázek 59	Stránka Databáze . . . . .	73
Obrázek 60	Formulář pro nový dataset . . . . .	73
Obrázek 61	Stránka Detail systému . . . . .	74
Obrázek 62	Detail planety . . . . .	74

Obrázek 63	Provnání velikostí . . . . .	75
Obrázek 64	Provnání vzdáleností . . . . .	75
Obrázek 65	Návod pro spuštění klientské aplikace . . . . .	76
Obrázek 66	Informace o procesu . . . . .	76
Obrázek 67	Přihlašovací formulář . . . . .	76
Obrázek 68	Profil a editace uživatele . . . . .	77

## SEZNAM TABULEK

Tabulka 1	Známé typy exoplanet . . . . .	20
Tabulka 2	Fáze oběhu tranzitující exoplanety . . . . .	23
Tabulka 3	Veličiny tranzitu . . . . .	23
Tabulka 4	Příklady výpočtu hmotnosti planet . . . . .	29
Tabulka 5	Komponenty formálního neuronu . . . . .	35
Tabulka 6	Parametry plně propojené vrstvy . . . . .	42
Tabulka 7	Parametry konvoluční vrstvy . . . . .	43
Tabulka 8	Parametry vrstvy Max-Pooling . . . . .	44
Tabulka 9	Parametry vrstvy Dropout . . . . .	45
Tabulka 10	Údaje o hvězdách ukládané do databáze . . . . .	67
Tabulka 11	Údaje o planetách ukládané do databáze . . . . .	68
Tabulka 12	Pojmenování soustavy Kepler-10 v různých katalozích . . . . .	68
Tabulka 13	Požadavky pro spuštění aplikace . . . . .	69
Tabulka 14	Výchozí adresy komponent po spuštění . . . . .	69
Tabulka 15	Autentizační komponenty . . . . .	77
Tabulka 16	Verifikace výsledků aplikace . . . . .	78

# SEZNAM ZDROJOVÝCH KÓDŮ

Zdrojový kód 1	Pseudokód genetického algoritmu . . . . .	50
Zdrojový kód 2	Ukázka práce s knihovnou React . . . . .	53
Zdrojový kód 3	Ukázka práce s knihovnou Styled components . . . . .	53
Zdrojový kód 4	Ukázka práce s microframeworkem Flask . . . . .	54
Zdrojový kód 5	Ukázka práce s knihovnou Astropy . . . . .	54
Zdrojový kód 6	Ukázka práce s knihovnou LightKurve . . . . .	55
Zdrojový kód 7	Tvorba neuronové sítě v Keras . . . . .	55
Zdrojový kód 8	Definice schématu v MongoEngine . . . . .	56
Zdrojový kód 9	Ukázka Aggregation Frameworku z MongoDB . . . . .	56
Zdrojový kód 10	Ukázka komunikace pomocí socket.io. . . . .	56
Zdrojový kód 11	Vytvoření modelů v REST API. . . . .	58
Zdrojový kód 12	Vytvoření koncového bodu v REST API. . . . .	58
Zdrojový kód 13	Konfigurace zdroje v REST API. . . . .	59
Zdrojový kód 14	Ukázka použití knihovny Redux. . . . .	61
Zdrojový kód 15	Příklad zpracování světelné křivky klientem . . . . .	63
Zdrojový kód 16	Použití sítě pro klasifikaci hvězdy . . . . .	65
Zdrojový kód 17	Připojení k databázi. . . . .	66
Zdrojový kód 18	Sestavení a spuštění aplikace (Linux). . . . .	69
Zdrojový kód 19	Výchozí konfigurace serveru. . . . .	69
Zdrojový kód 20	Spuštění a sestavení aplikace s vlastní konfigurací (Linux). . . . .	70
Zdrojový kód 21	Výchozí konfigurace webové aplikace. . . . .	70
Zdrojový kód 22	Argumenty spouštěcích skriptů (Linux). . . . .	70
Zdrojový kód 23	Spuštění v režimu pro vývoj (Linux). . . . .	70
Zdrojový kód 24	Spuštění testů aplikace (Linux). . . . .	71

# SEZNAM VZORCŮ

1	Pravděpodobnost zpozorování tranzitu planety přes hvězdu . . . . .	22
2	Výpočet velké poloosy dráhy planety . . . . .	25
3	Výpočet poloměru planety . . . . .	25
4	Odhad průměrné rychlosti oběhu planety . . . . .	26
5	Výpočet inklinace dráhy . . . . .	26
6	Radiální rychlost na základě změny vlnové délky . . . . .	28
7	Vztah radiální rychlosti a hmotnosti planety . . . . .	28
8	Poměr zářivého výkonu hvězdy a planety . . . . .	30
9	Funkce vyjadřující přesnost modelu vůči realitě při počítání TTV . . . . .	33
10	Rovnice problému 3 těles . . . . .	34
11	Obecný vztah vstupu neuronu a jeho váhy . . . . .	36
12	Lineární vztah vstupu neuronu a jeho váhy . . . . .	37
13	Agregační funkce neuronu . . . . .	37
14	Hyperbolicko-tangenciální aktivační funkce . . . . .	37
15	Sigmoidální aktivační funkce . . . . .	37
16	Gaussova aktivační funkce . . . . .	37
17	Provádění konvoluce pro výřez vstupu a filtr . . . . .	43
18	Výpočet chyby pro jeden vzor u algoritmu zpětného šíření chyby . . . . .	46
19	Přírůstek váhy u algoritmu zpětného šíření chyby . . . . .	46
20	Výpočet lokálního gradientu neuronu . . . . .	47

## SEZNAM ZKRATEK

AI	Artificial intelligence
API	Application Programming Interface
au	Astronomical unit
CNN	Convolutional neural network
CRUD	Create, Read, Update, Delete
csv	Comma-separated values
FC	Fully-connected layer
FFNN	Feed-forward neural network
fits	Flexible image transport system
GA	Genetic algorithm
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSX	JavaScript XML
ly	Light year
NASA	National Aeronautics and Space Administration
ODM	Object-document mapper
pc	Parsec
REST	Representational state transfer
TESS	Transiting Exoplanet Survey Satellite
TPF	Target pixel file
TS	TypeScript
TSP	Travelling salesman problem
TSX	TypeScript XML
TTV	Transit timing variation
UI	User interface
URL	Uniform Resource Locator
XML	Extensible Markup Language



# ÚVOD

V naší sluneční soustavě se nachází celkem 8 dosud objevených planet včetně Země. Mimo ni ale v pozorovatelném vesmíru existují odhadem stovky miliard galaxií a v každé z nich v průměru stovky miliard hvězd. Z toho, co o vzniku a fungování hvězdných soustav víme je pravděpodobné, že většinu těchto hvězd bude obíhat jedna nebo více planet, tzv. extrasolárních planet nebo také exoplanet. [12]

První potvrzená exoplaneta byla objevena již roku 1992, ale výzkum exoplanet se dostal do oblasti širokého zájmu až během posledního desetiletí. Stalo se tak především kvůli vesmírnému teleskopu Kepler, který má na svém kontě od roku 2009 přes 2 500 objevených exoplanet. [14, 25]

K dnešnímu dni je známo více jak 4 000 potvrzených exoplanet. Toto číslo se s nejvyšší pravděpodobností bude rychle zvyšovat, protože roku 2018 byl vypuštěn nástupce Kepleru – satelit TESS – od něhož je očekáván objev 20 000 exoplanet. [27]

Planety u jiných hvězd většinou nelze pozorovat přímo. Proto je nepřímými metodami zkoumáno jejich působení na své mateřské hvězdy, které už pozorovat lze. Výstupem z takovýchto pozorování jsou často stovky GiB fyzikálních a statistických dat, jež je následně nutno zpracovat. [12]

Cílem této diplomové práce je vytvořit aplikaci umožňující uživatelům poskytovat výpočetní výkon svých počítačů pro analýzu právě těchto dat. Projekt sestává z klientského programu, webové aplikace a serveru. Klientský program provádí potřebné distribuovatelné výpočty na počítači uživatele. Tento program je možné ovládat z rozhraní webové aplikace, jež zároveň poskytuje přehled o všech aktivitách, uživatelích a datech. Rozdělování výpočetních úloh mezi klienty a ukládání dat do databáze pak řeší server.

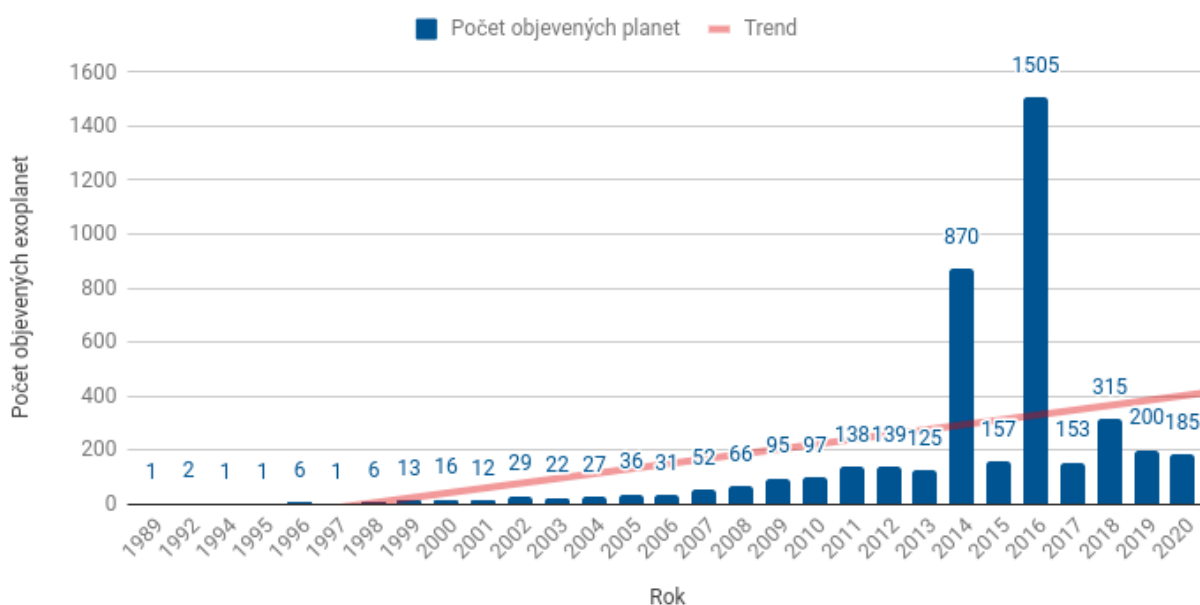
Díky distribuovaným výpočtům se do výzkumu exoplanet bude moci bez vysokého úsilí, znalosti či technického vybavení zapojit i široká veřejnost. To může urychlit vývoj a zároveň zvýšit povědomí o této vědní disciplíně.

V projektu jsou využity některé techniky spadající pod umělou inteligenci, v důsledku čehož je zpracovávání dat zcela automatizované. Platí však, že umělá inteligence je v současnosti stále intenzivně se rozvíjející oblastí, a proto výsledky nemusí být natolik vypovídající ve srovnání s tím, kdy by výzkum prováděli lidé manuálně, byť by to trvalo nesrovnatelně déle.

# 1 HLEDÁNÍ EXOPLANET

Pouhým zkoumáním planet v naší sluneční soustavě jsme omezeni na velice specifické podmínky existující v okolí našeho Slunce. Pro hlubší pochopení fungování planetárních systémů je nutné rozšířit oblast zájmu i na planety v okolí jiných hvězd – tzv. exoplanety. Je tak možno se přiblížit odpovědím na otázky jako „Jak vzácné jsou podmínky pro život ve vesmíru?“ nebo „Jak vznikla a jak se vyvíjela naše planeta?“ [12]

Na počátku 90. let minulého století byla objevena první exoplaneta v okolí pulsaru a v roce 1995 první exoplaneta v okolí hvězdy podobné Slunci. Od té doby frekvence objevů planet v průměru neustále stoupá. [12, 25]

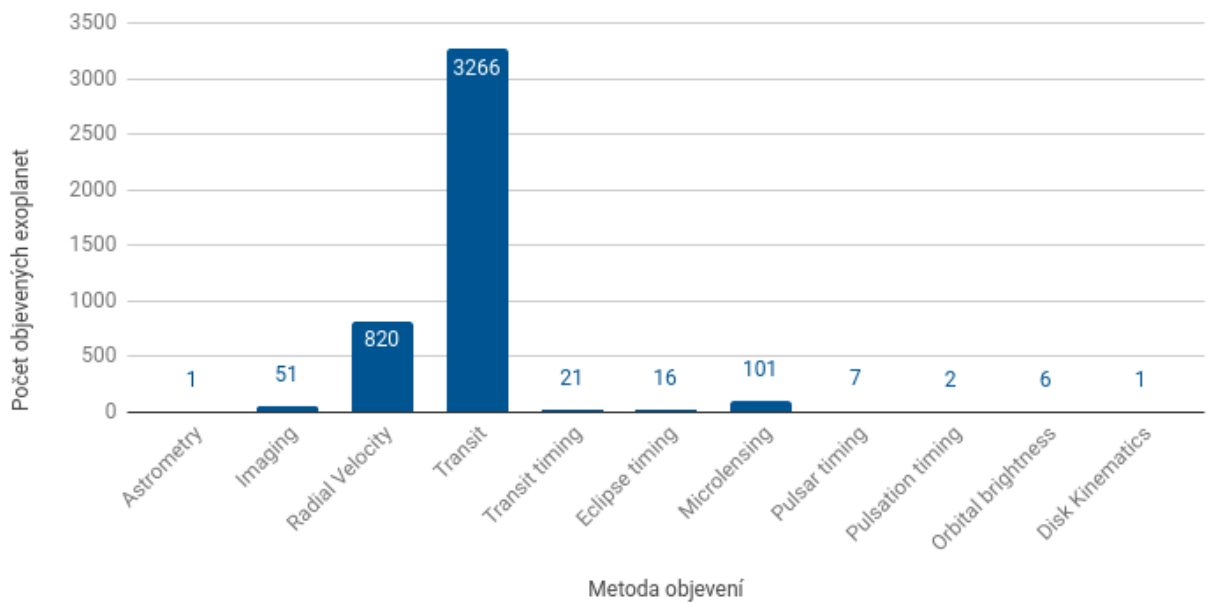


Obrázek 1: Četnost objevů exoplanet v jednotlivých letech <sup>1</sup>

Cestovat k jiným hvězdám a posílat sondy k exoplanetám je však naprosto mimo možnosti naší současné technologie. Dokonce i přímé pozorování exoplanet teleskopem je ve většině případů nemožné. Téměř veškeré objevy jsou prováděny skrze nepřímé metody. Ty využívají skutečnosti, že i když není možné spatřit exoplanetu samotnou, je možné detekovat její působení na své okolí (např. na mateřskou hvězdu).

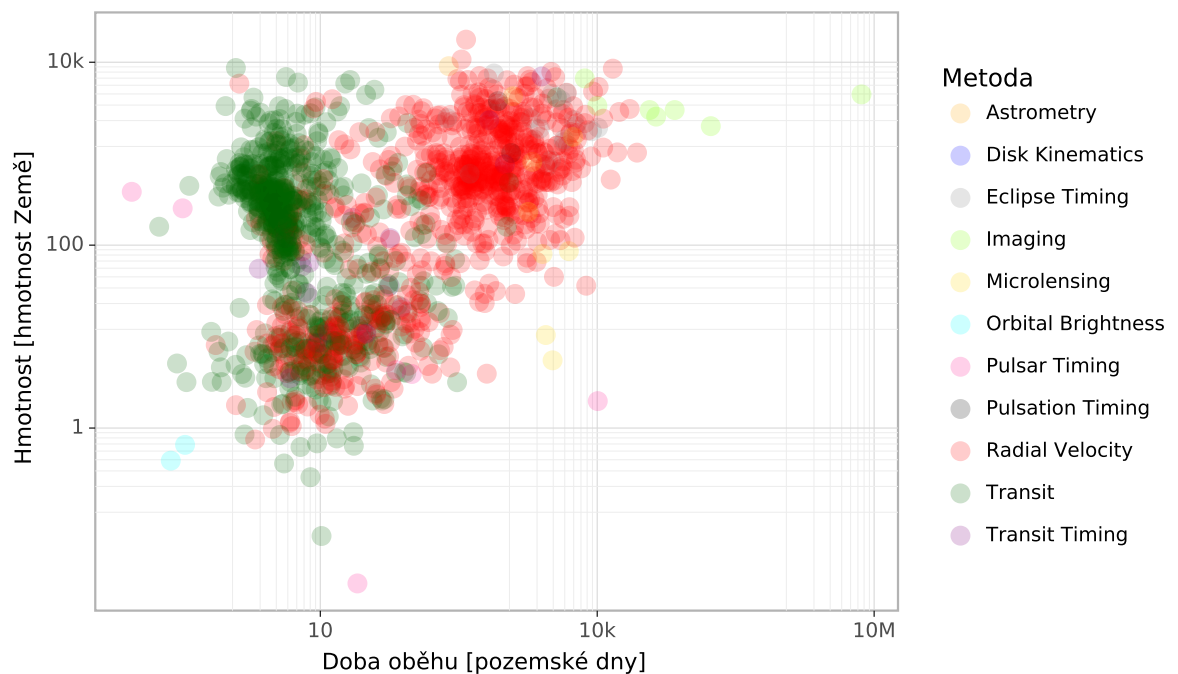
Jednotlivé metody budou popsány v následujících podkapitolách. Zdaleka nejvýznamnější je tranzitní metoda, kterou byla objevena většina exoplanet. Velké množství planet bylo objeveno taktéž metodou radiálních rychlostí. [25]

<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].



Obrázek 2: Počty objevených exoplanet jednotlivými metodami <sup>1</sup>

Dá se říci, že každá metoda je vhodnější pro objevování různých typů planet. Málo hmotné planety byly objevovány častěji tranzitní metodou, zatímco hmotnější planety spíše metodou radiálních rychlostí. Pro planety vzdálené od své mateřské hvězdy se nejlépe osvědčila metoda přímého zobrazení. [25]



Obrázek 3: Známé exoplanety dle jejich vlastností a metody objevení <sup>1</sup>

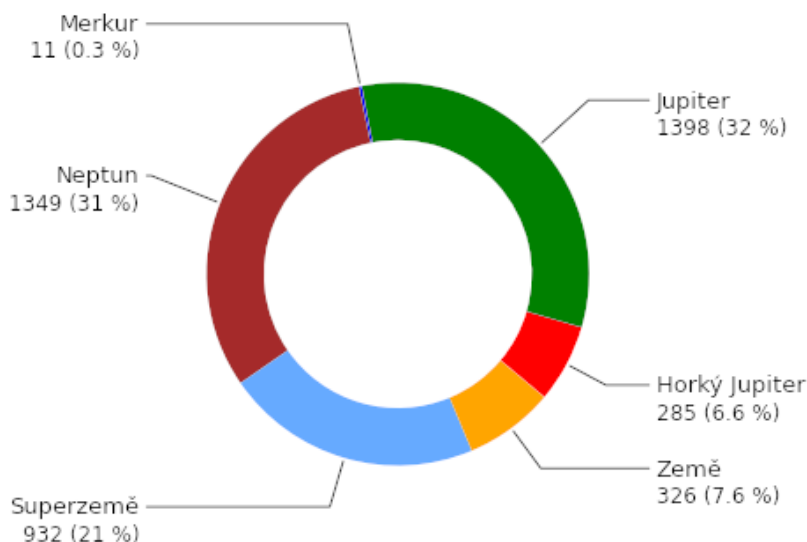
<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].

Na základě známých charakteristik lze jednotlivé planety zařadit do jedné z následujících kategorií:

Typ	Popis	V naší soustavě
Podobná Merkuru	Malé kamenné planety. Vzhledem k zanedbatelnému působení těchto planet na okolí je velice těžké je detekovat.	Merkur, Mars
Exo-Země	Kamenné planety velikostně podobné Zemi. V žádném případě se nejedná o planety s garantovanými podmínkami pro život. Přesto však ze všech typů planet představují nejvyšší šanci na nalezení těchto podmínek.	Země, Venuše
Podobná Neptunu	Planety, jejichž atmosféra je tvořena převážně vodíkem a heliem s jádrem z těžkých kovů. Velikostně podobné Neptunu.	Neptun, Uran
Plynný obr	Velké plynné planety podobně velké nebo i větší než Jupiter.	Jupiter, Saturn
Superzemě	Planety větší než Země, ale menší než Neptun. Dosahují až 10násobku hmotnosti Země. Může se jednat jak o kamenné planety, tak o vodní či ledové světy nebo i plynné útvary. Ty se označují jako sub-Neptun nebo mini-Neptun.	<i>Neexistuje</i>
Horký Jupiter	Zvláštní typ plynných obrů, které narozdíl od těch v naší soustavě obíhají v těsné blízkosti své hvězdy. Jsou rozpálené na tisíce K a mohou významně působit na svou hvězdu. Díky tomu jsou první objevené exoplanety právě horké Jupitery.	<i>Neexistuje</i>

Tabulka 1: Známé typy exoplanet <sup>1</sup>

Nejvíce objevených exoplanet spadá do kategorie plynných obrů, které kvůli vysoké hmotnosti i velikosti značně ovlivňují svou hvězdu. Detekovat působení malých kamenných planet je náročnější, a proto je těchto exoplanet objeveno naopak nejméně.

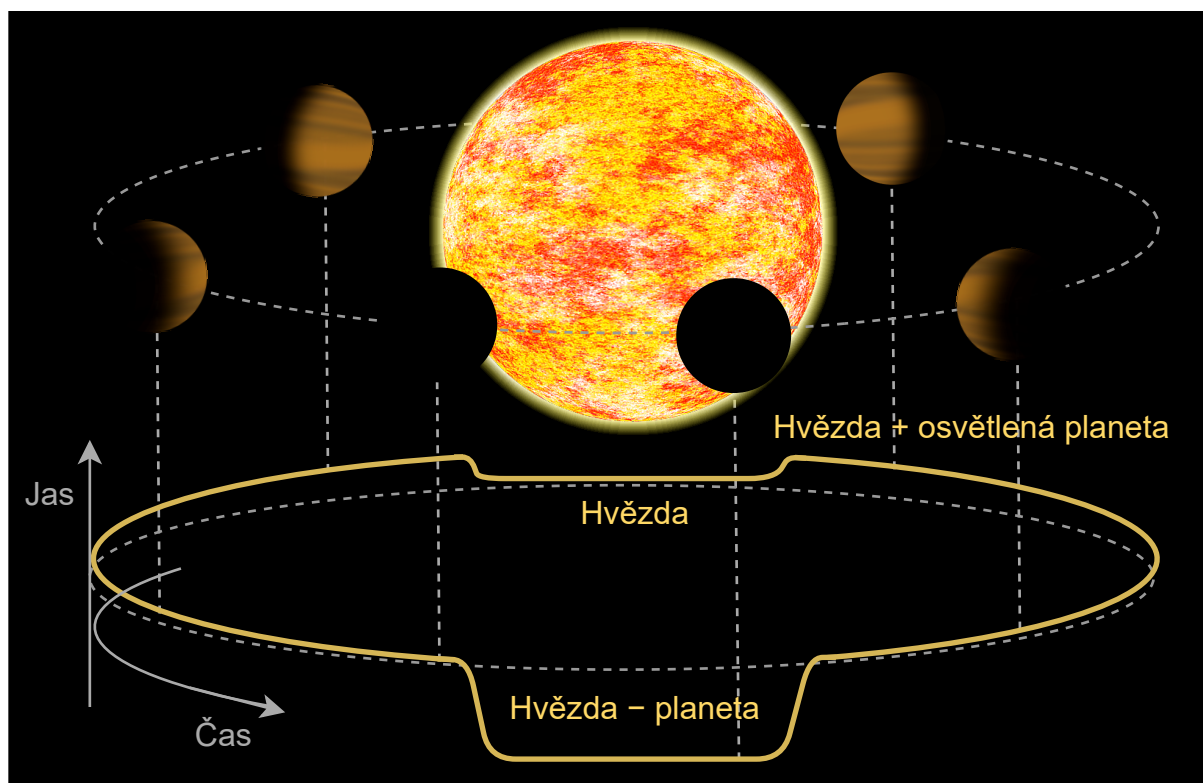


Obrázek 4: Počty objevených exoplanet dle jejich typu <sup>1</sup>

<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].

## 1.1 Tranzitní metoda

Někdy se planeta při obíhání dostane mezi svou hvězdou a Zemí. Tento jev se pro pozorovatele na Zemi projeví jako mírný pokles jasu hvězdy (obvykle ve zlomku procenta). Při dlouhodobém pozorování je možné v těchto změnách jasu hvězdy odhalit opakující se složku. To by mohlo indikovat přítomnost planety v blízkosti této hvězdy. [12, 8]



Obrázek 5: Přejchod planety přes kotouč hvězdy <sup>1</sup>

Tyto změny však nemusí být na první pohled viditelné, protože v soustavě může být více planet, které svou hvězdu zastiňují různou měrou a obíhají kolem ní s různou periodou. Navíc i v situaci, kdy je ve změnách jasu hvězdy objevena periodická složka nemusí jít vždy o obíhající planetu. Hvězda může být např. sama o sobě proměnlivá nebo se může jednat o dvojhvězdu, jejíž složky se vzájemně zastiňují. [14]

Tranzitní metoda vzbuzuje velký zájem především kvůli možnosti objevovat i malé planety podobné Zemi – takové planety by mohly spíše splňovat podmínky pro život. Nevýhodou je, že většina exoplanet obíhá svou hvězdu v takové rovině, v jaké pozorovatel na Zemi nemůže transit spatřit. Odhadem 99 % všech potenciálních exoplanet podobných Zemi nemůže být tranzitní metodou nikdy zachyceno. [22, 8]

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io> a GIMP.

$$P = \frac{d_s}{a} \quad (1)$$

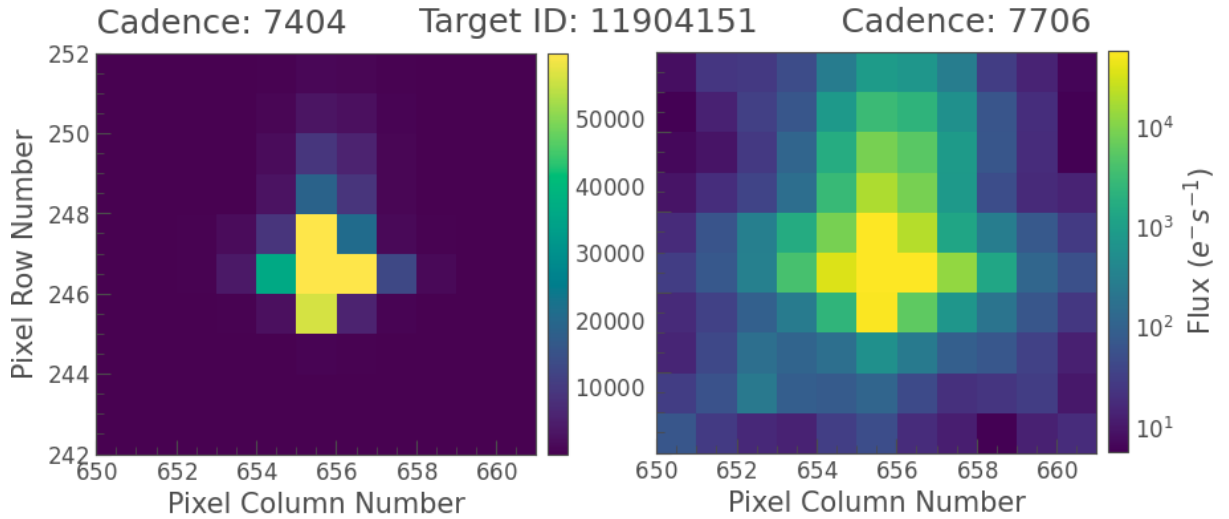
Vzorec 1: Pravděpodobnost zpozorování tranzitu planety přes hvězdu

$d_s$  = průměr hvězdy     $a$  = vzdálenost exoplanety od hvězdy

### 1.1.1 Target pixel file

Prvním krokem v analýze hvězdy tranzitní metodou je její fyzické pozorování. Teleskop obvykle pozoruje část oblohy po dobu několika měsíců, přičemž každých několik desítek minut vytvoří snímek dané části oblohy. Z výsledných fotografií se následně vyextrahují jednotlivé hvězdy, čímž vzniknou tzv. target pixel files.

TPF obsahuje část oblohy o velikosti několika pixelů, na které se v původní fotografii nacházela zkoumaná hvězda a její okolí. Barva pixelů je určena jasem.

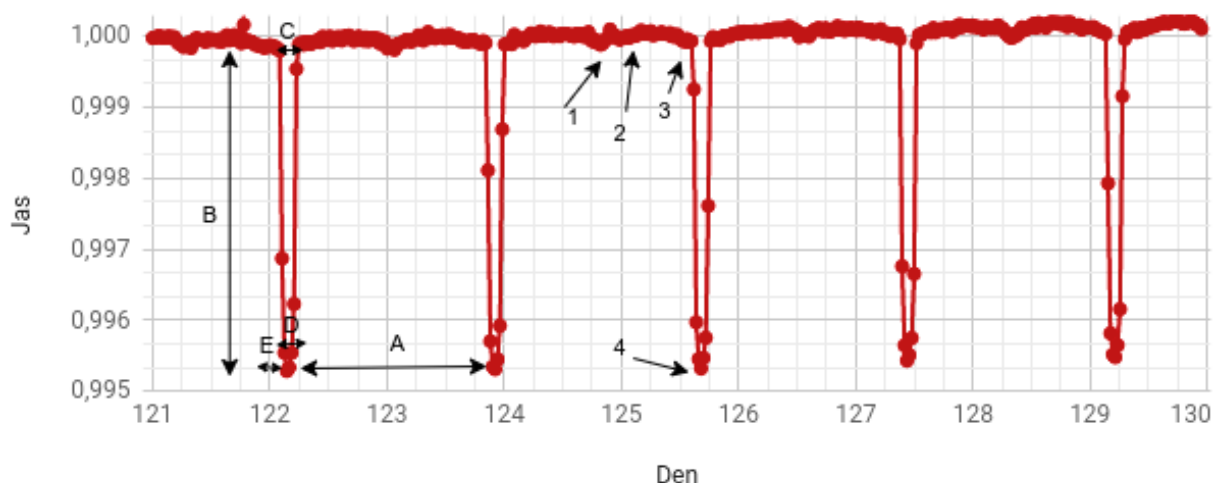


Obrázek 6: TPF soustavy Kepler-10 s lineárním a logaritmickým měřítkem <sup>1</sup>

### 1.1.2 Světelná křivka hvězdy

Po složení všech TPF do časové řady a vypočítání jejich jasů vznikne světelná křivka. Na obrázku 5 je světelná křivka hvězdy Kepler-13 očištěná od dlouhodobého trendu, šumu a extrémních hodnot. Křivka vykazuje velice výraznou periodickou složku s periodou 1,763 dne. Ve většině případů ale vliv planety není takto výrazný a detekovat planetu je obtížnější.

<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].



Obrázek 7: Světelná křivka soustavy Kepler-13 <sup>1</sup>

Na obrázku 6 jsou čísla označeny jednotlivé fáze dále popsané v tabulce 2:

Fáze	Co vidí pozorovatel na Zemi
1 Planeta je za hvězdou (sekundární zákryt).	Hvězda
2 Planeta je vedle hvězdy.	Hvězda + osvětlená část planety
3 Planeta je vedle hvězdy.	Hvězda + neosvětlená část planety
4 Planeta je před hvězdou (tranzit).	Hvězda – planeta

Tabulka 2: Fáze oběhu tranzitující exoplanety

Dále jsou na stejném obrázku písmeny označeny důležité veličiny:

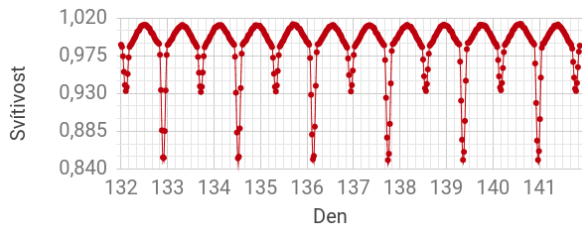
Veličina	Popis
A Perioda	Perioda tranzitu udává periodu oběhu planety kolem hvězdy.
B Hloubka	Čím větší je hloubka tranzitu, tím je planeta vůči hvězdě větší.
C Trvání	Čím je trvání delší, tím delší trajektorii přes hvězdu planeta má.
D Trvání minima	Závisí na úhlu mezi rovinou oběhu planety vůči pozorovateli
E Trvání nástupu	

Tabulka 3: Veličiny tranzitu

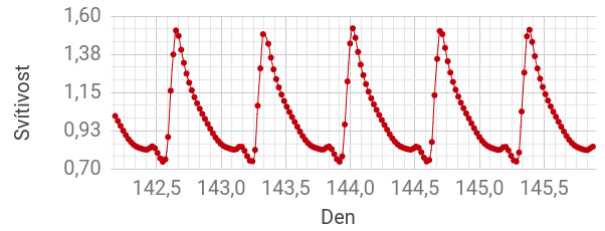
### 1.1.3 Vyřazení falešně pozitivních výsledků

Většina periodických složek ve světelných křivkách hvězd jsou *false positive* – patří jiným jevům, než je obíhající planeta. Tyto případy je třeba odfiltrovat, což byla až donedávna především manuální práce lidí – vědců či dobrovolníků. Protože ale tranzit planety vykazuje specifický průběh popsany v předchozí kapitole, je možné ho s určitou úspěšností rozpoznat pomocí naučené umělé neuronové sítě automaticky. [14]

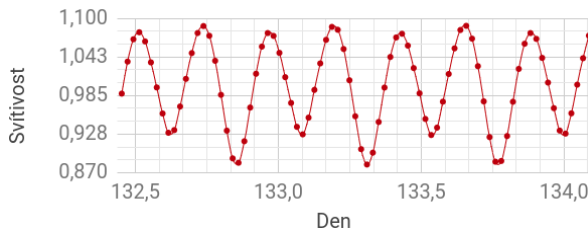
<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].



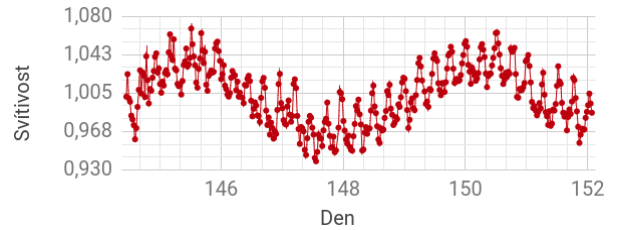
Obrázek 8: Dvojhvězda KIC 8262223 <sup>1</sup>



Obrázek 9: Cefeida KIC 3733346 <sup>1</sup>

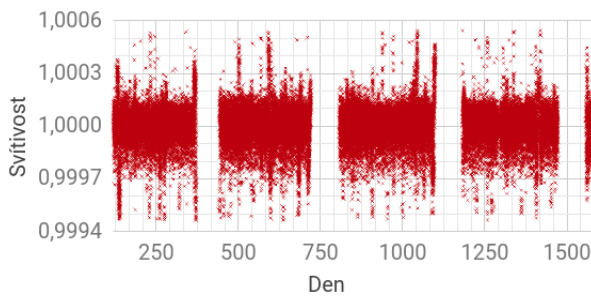


Obrázek 10: Proměnná hvězda KIC 9832227 <sup>1</sup>

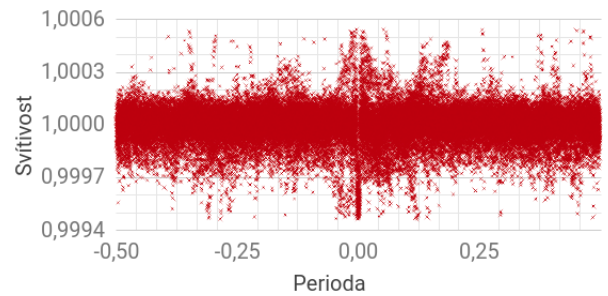


Obrázek 11: Kataklyzmická proměnná hvězda KIC 9406652 <sup>1</sup>

Je třeba aby vstupní data do neuronové sítě měla stejné rozměry i formát. Vzhledem k různorodosti světelných křivek je nutno provést několik kroků, abychom dosáhli standardizovaného formátu. Prvním krokem je složení časové řady do jedné periody, čímž dojde k posílení viditelnosti transitu (pokud zde nějaký je), nebo naopak k jeho vyrušení (pokud zde žádný není). [14]



Obrázek 12: Světelná křivka Kepler-10 <sup>1</sup>



Obrázek 13: Složená světelná křivka <sup>1</sup>

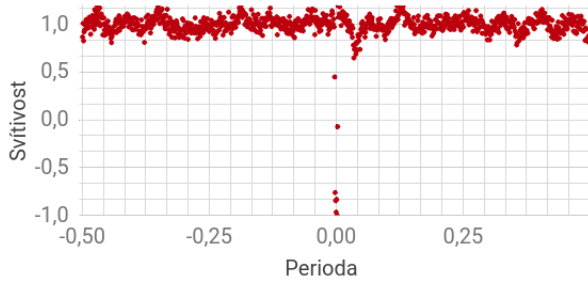
Na obrázku 13 je uprostřed slabě patrný transit. Má malou šířku, protože trvá pouze 0,25 dne, zatímco celá perioda je dlouhá 45,3 dne. Z této složené časové řady se vytvoří dva pohledy, které budou vstupem do neuronové sítě:

- **Globální pohled** (*obr. 14*) – Šířka periody a počet bodů v časové řadě jsou fixní. Nevýhodou je, že u planet s dlouhou periodou bude transit velice nepatrný, proto pouze globální pohled nestačí. [14]

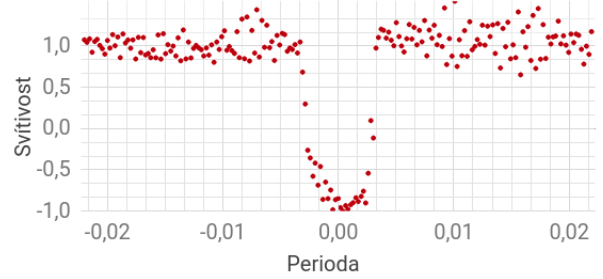
<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].



- **Lokální pohled** (obr. 15) – Šířka tranzitu a počet bodů v časové řadě jsou fixní. Nevýhodou je, že není viditelný celý průběh světelné křivky. Naproti tomu je ale zřetelný tranzit. [14]



Obrázek 14: Globální pohled na tranzit Kepler-10 c <sup>1</sup>



Obrázek 15: Lokální pohled na tranzit Kepler-10 c <sup>1</sup>

Fixní počet bodů lze zajistit nahrazením každých  $\frac{\alpha}{\beta}$  sousedních bodů ( $\alpha$  – současný počet bodů,  $\beta$  – požadovaný počet bodů) jediným, který bude reprezentovat jejich medián. Pro potřeby neuronové sítě je vhodné oba pohledy taktéž normalizovat tak, aby platilo  $H = \langle 1; -1 \rangle$ . U případů, které neuronová síť vyhodnotí jako planety, je možné pokračovat výpočtem dalších informací o planetě. [14]

### 1.1.4 Výpočet vlastností planety

Velkou poloosu dráhy planety lze vypočítat s využitím hmotnosti hvězdy a periodu oběhu z třetího Keplerova zákona. [10]

$$a = \sqrt[3]{\frac{GM P^2}{4\pi^2}} \quad (2)$$

Vzorec 2: Výpočet velké poloosy dráhy planety

$a$  = velká poloosa  $G$  = gravitační konstanta  $M$  = hmotnost hvězdy  $P$  = perioda oběhu planety

Ze světelné křivky a poloměru hvězdy je možno vypočítat poloměr tranzitující planety po vyjádření z následující rovnice [8, 10]:

$$\frac{r^2}{R^2} = \frac{\Delta F}{F} \quad (3)$$

Vzorec 3: Výpočet poloměru planety

$r$  = poloměr planety  $R$  = poloměr hvězdy  $F$  = jas hvězdy  $\Delta F$  = změna jasu

Dále lze odhadnout i průměrnou rychlost pohybu planety po oběžné dráze. Skutečná průměrná rychlost však může být jiná, protože se nepočítá s excentricitou dráhy [10]:

<sup>1</sup> Vytvořeno autorem, zdroj dat: <https://exoplanetarchive.ipac.caltech.edu>.

$$v \approx \frac{2\pi a}{T} \quad (4)$$

Vzorec 4: Odhad průměrné rychlosti oběhu planety

$v$  = rychlost oběhu planety    $a$  = velká poloosa dráhy planety    $T$  = perioda oběhu planety

Další z důležitých charakteristik orbity je inklinace (sklon), která nám řekne, jaký úhel svírá rovina oběhu exoplanety vůči pozorovateli. Bohužel lze zjistit pouze minimální, nikoliv skutečnou velikost úhlu sklonu dráhy. Inklinace se bude zpravidla blížit  $90^\circ$ . [10]

$$\cos i \leq \frac{R + r}{a} \quad (5)$$

Vzorec 5: Výpočet inklinace dráhy

$i$  = inklinace    $R$  = poloměr hvězdy    $r$  = poloměr planety    $a$  = velká poloosa dráhy planety

### 1.1.5 Výpočet hmotnosti planety

Tranzitní metoda nenabízí analytický způsob výpočtu hmotnosti exoplanety. Na základě již známých dat lze pozorovat korelaci mezi různými veličinami. Např. čím je planeta větší, tím pravděpodobněji bude i hmotnější. Není to však pravidlem. Jistější je spoléhat se na korelaci mezi více veličinami. Nalézt vztahy ve vícerozměrných datech však nemusí být vždy jednoduché. A zde nachází uplatnění umělá neuronová síť, která je mimo jiné určena pro hledání právě takovýchto vztahů. [15]

Vše, co je třeba, je dostatečně velký dataset s planetami, u kterých jsou známy všechny potřebné veličiny. Na těchto datech proběhne trénování neuronové sítě, která se naučí hustotu rozdělení pravděpodobností hodnot výše zmíněných veličin. S takto naučenou sítí lze dále provádět 2 typy úloh:

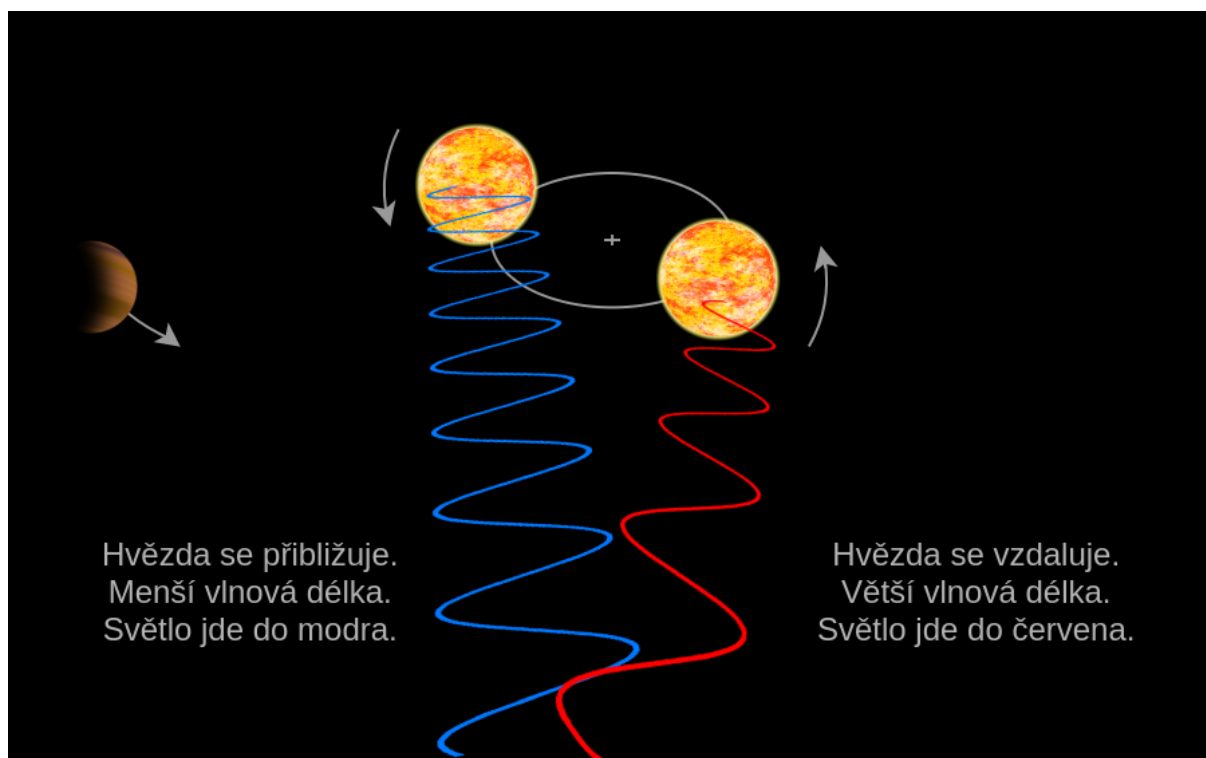
- Vygenerovat novou „umělou“ planetu, jejíž vlastnosti jsou realistické (odpovídají náhodným rozdělením planet v trénovací množině),
- Nebo na základě známých veličin planety odhadnout veličinu neznámou.

Právě druhý ze zmíněných typů úloh lze použít pro výpočet hmotnosti. Např. na základě poloměru, periody a rovnovážné teploty planety, které jsou tranzitní metodou lehce spočitatelné, je možno odhadnout chybějící hmotnost planety. [15]

## 1.2 Metoda radiálních rychlostí

Stejně jako hvězda ovlivňuje obíhající planetu, tak i planeta gravitačně ovlivňuje svou hvězdu a obě tělesa obíhají kolem společného těžiště. Tento pohyb se může projevit jako opakované přibližování a vzdalování hvězdy vůči pozorovateli na Zemi. Právě pojem radiální rychlost označuje rychlost pohybu ve směru k pozorovateli. [22]

Pokud se zdroj elektromagnetického záření (hvězda) přibližuje vůči pozorovateli, záření má menší vlnovou délku a jeví se více do modra, protože právě modrá (a fialová) barva má z viditelného spektra nejmenší vlnovou délku. Obdobná situace nastává při vzdalování se zdroje vlnění od pozorovatele. Vlnová délka se zvětšuje a barva jde do červena. Tomuto efektu se říká červený (resp. modrý) posuv. [22]



Obrázek 16: Metoda radiálních vzdáleností <sup>1</sup>

Příčinou červeného/modrého posuvu je v tomto případě Dopplerův jev, který lze uplatnit i pro jiné druhy vlnění, než to elektromagnetické – zvuk. Pokud se k nám zdroj zvuku přibližuje (např. siréna na jedoucím autě), zvuk má zpravidla vyšším tón, protože má menší vlnovou délku (vyšší frekvenci). Při vzdalování zdroje má zvuk větší vlnovou délku a je naopak vnímán hlubším tónem. [6] Periodicky se opakující změny ve vlnové délce záření hvězdy tak mohou být důsledkem existence tělesa v této soustavě. [22]

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io> a GIMP.

### 1.2.1 Výpočet radiální rychlosti hvězdy

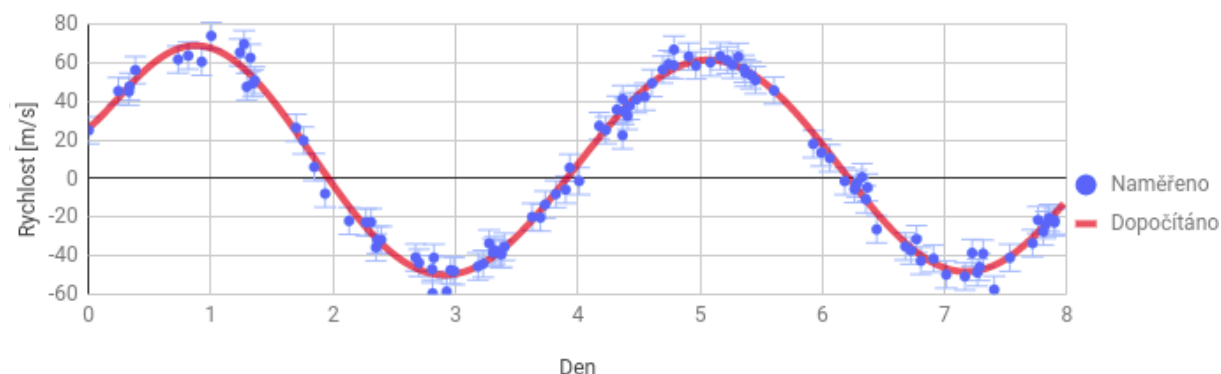
Poté, co teleskop sesbírá dostatečně velkou časovou řadu vlnových délek záření hvězdy může dojít k vypočítání radiální rychlosti v čase. Platí, že radiální rychlost je kladná, pokud se zdroj od pozorovatele vzdaluje a záporná pokud se přibližuje. [22]

$$v = c * \frac{\Delta\lambda}{\lambda_0} \quad (6)$$

Vzorec 6: Radiální rychlost na základě změny vlnové délky

$\Delta\lambda$  = změna vlnové délky  $\lambda_0$  = klidová vlnová délka  $v$  = radiální rychlost

Graf znázorňující radiální rychlost hvězdy 51 Pegasi může vypadat takto:



Obrázek 17: Radiální rychlost hvězdy 51 Pegasi v čase <sup>1</sup>

Na první pohled je patrná jedna periodická složka s periodou 4,23 dne značící, že kolem této hvězdy obíhá jedna planeta. Aby metoda radiálních rychlostí fungovala, musí být exoplaneta vůči své hvězdě dostatečně hmotná a také nesmí obíhat po rovině, která je kolmá k přímce směrem k pozorovateli.

### 1.2.2 Výpočet hmotnosti planety

Hlavní výhodou metody radiálních rychlostí oproti tranzitní metodě je možnost spočítat hmotnost exoplanety. Její přesnou hodnotu však lze odvodit pouze se znalostí sklonu dráhy exoplanety vůči pozorovateli. V opačném případě lze vypočítat pouze dolní mez hmotnosti planety, a to zejména kvůli  $M_p * \sin(i)$  ve vzorci 7. [22]

$$\Delta v_{max} = \sqrt[3]{\frac{2\pi G}{T}} * \frac{M_p * \sin(i)}{\sqrt[3]{(M_p + M_s)^2}} * \frac{1}{\sqrt{1 - e^2}} \quad (7)$$

Vzorec 7: Vztah radiální rychlosti a hmotnosti planety

$\Delta v_{max}$  = amplituda změny rychlosti  $M_s$  = hmotnost hvězdy  $M_p$  = hmotnost planety  
 $\sin(i)$  = sklon dráhy vůči pozorovateli  $e$  = excentricita dráhy  $T$  = oběžná doba

<sup>1</sup> Vytvořeno autorem, zdroj dat: [7].

Těleso	Hvězda	$\Delta v_{max}$ [ $\frac{m}{s}$ ]	$M_p$ [kg]	$\sin(i)$	e	T [r]	$M_s$ [kg]
Země	Slunce	0,089	$2 \cdot 10^{30}$	1	0,017	1	$5,97 \cdot 10^{24}$
Jupiter		12,4			0,048	11,86	$1,9 \cdot 10^{27}$
Pluto		0,00003			0,247	247,41	$1,3 \cdot 10^{22}$
$\alpha$ Cen Bb	$\alpha$ Cen B	0,51	$1,8 \cdot 10^{30}$		0	0,0089	$6,75 \cdot 10^{24}$
51 Pegasi b	51 Pegasi	55,9	$2,22 \cdot 10^{30}$		0,013	0,0116	$0,88 \cdot 10^{27}$

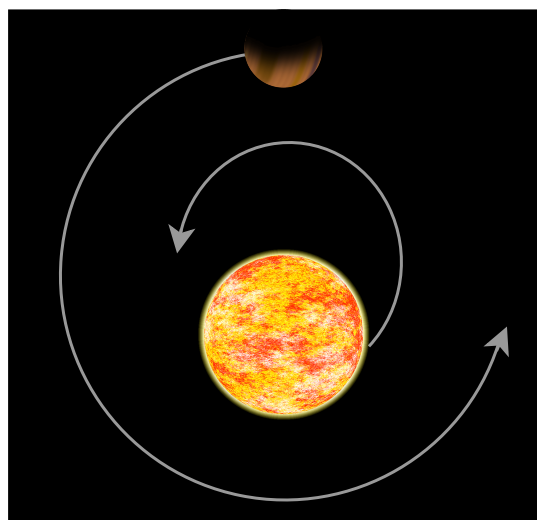
Tabulka 4: Příklady výpočtu hmotnosti planet <sup>1</sup>

### 1.3 Astrometrická metoda

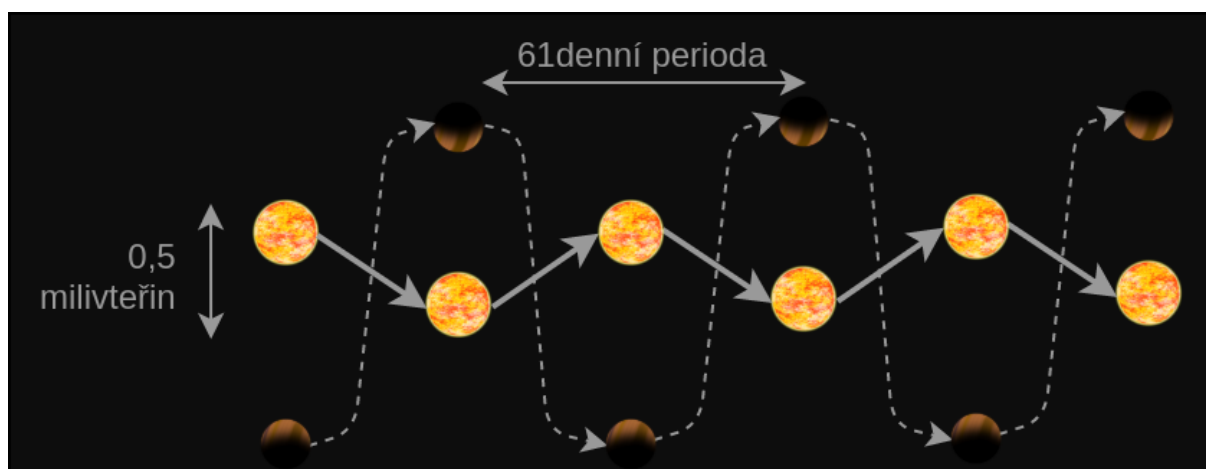
Astrometrická metoda využívá stejné vlastnosti vzájemného působení těles jako metoda radiálních rychlostí. Namísto zkoumání vlnové délky záření se však zaměřuje na polohu hvězdy. Hvězda, kolem níž obíhá dostatečně hmotné těleso, se bude v důsledku jeho působení nepatrně vychylovat ze své pozice. [22]

Pohyb hvězdy tak není přímočarý, ale vlnitý. Kolísání hvězdy je však pro pozorovatele na Zemi často pouze v řádu stovek úhlových mikrovteřin. [22] Z tohoto důvodu byla astro-

metrickou metodou dosud objevena pouze jediná exoplaneta. Dá se však očekávat, že se zlepšující se technikou bude tato metoda úspěšnější. [25]



Obrázek 18: Obíhání hvězdy a planety kolem společného těžiště <sup>2</sup>



Obrázek 19: Kolísání hvězdy Gliese 876 s planetou <sup>2</sup>

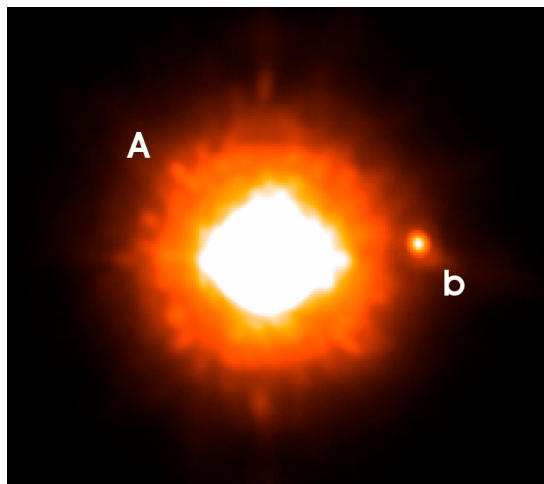
<sup>1</sup> Vytvořeno autorem, zdroj dat: [6, 7, 22, 25].

<sup>2</sup> Vytvořeno autorem v <https://www.draw.io> a GIMP.

## 1.4 Přímé zobrazení

Přímé pozorování planety v okolí hvězd je velice náročnou technikou, protože záření, které hvězda emituje je mnohonásobně silnější, než záření, které planeta odráží. Např. u Země a Slunce je tento poměr řádově  $10^{-9}$ . [22]

Největším problémem je zpravidla vliv atmosféry – chvění obrazu v důsledku vzájemného působení teplého a studeného vzduchu. Tento vliv je nutno potlačit, a to buď použitím adaptivní optiky, nebo pozorováním z vesmíru. [22]



Obrázek 20: Hvězda GQ Lupi s přímo pozorovanou exoplanetou <sup>1</sup>

$$\frac{L_p}{L_s} = \alpha \left( \frac{R_p}{a} \right)^2 \quad (8)$$

Vzorec 8: Poměr zářivého výkonu hvězdy a planety

$\alpha$  = odrazivost povrchu planety (albedo) |  $R_p$  = poloměr planety  
 $a$  = velká poloosa oběžné dráhy planety

## 1.5 Gravitační mikročochky

Světlo má obvykle tendenci se pohybovat prázdným prostorem po přímé trajektorii. Avšak dle obecné teorie relativity, hmotná tělesa zakřivují časoprostor kolem sebe. V blízkosti takových těles je tato přímá trajektorie zakřivená. [22]

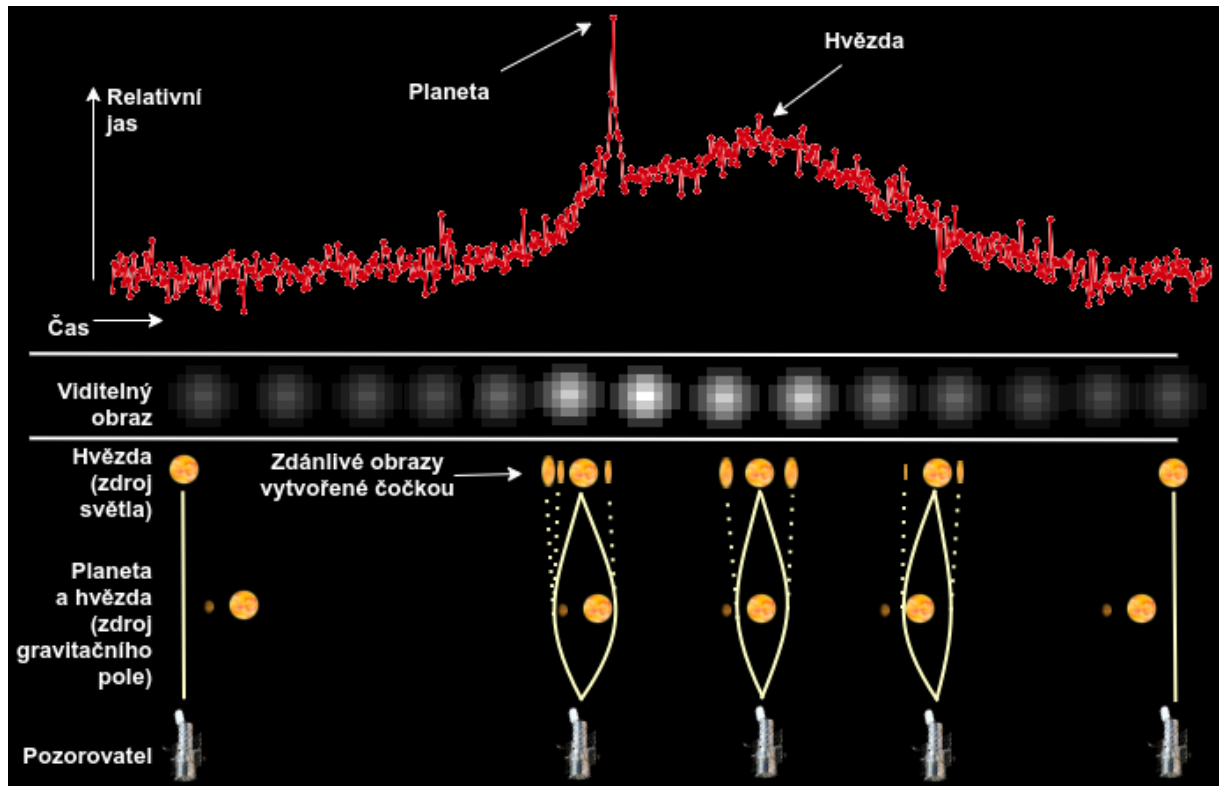


Obrázek 21: Princip gravitační čočky <sup>2</sup>

<sup>1</sup> Převzato z: [22].

<sup>2</sup> Vytvořeno autorem, fotografie gravitační čočky převzata od NASA ([https://apod.nasa.gov/apod/image/1112/lensshoe\\_hubble\\_3235.jpg](https://apod.nasa.gov/apod/image/1112/lensshoe_hubble_3235.jpg)), oříznuto a zdeformováno

V případě, kdy se nějaké hmotné těleso (např. galaxie) nachází mezi pozorovatelem a zdrojem světla (např. jinou galaxií), jedná se o tzv. gravitační čočce – zdroj světla bude viděn vícekrát na různých místech, nebo bude naopak zdeformován. [22]



Obrázek 22: Ilustrace gravitační mikročocky OGLE 2003-BLG-235 <sup>1</sup>

Pokud je čočkujícím tělesem pouze hvězda, jedná se o gravitační mikročocku. V tomto případě dochází často ke splnutí čočky a skrytého tělesa do jednoho útvaru se zvýšeným jasem. A zde nachází uplatnění metoda pro hledání exoplanet. Za předpokladu, že čočkujícím tělesem je hvězda s obíhající exoplanetou, lze pozorovat pozvolné zvyšování jasu způsobené hvězdou a zároveň v nějakém krátkém časovém intervalu strmý nárůst jasu způsobený exoplanetou. [22]

## 1.6 Časování pulsarů

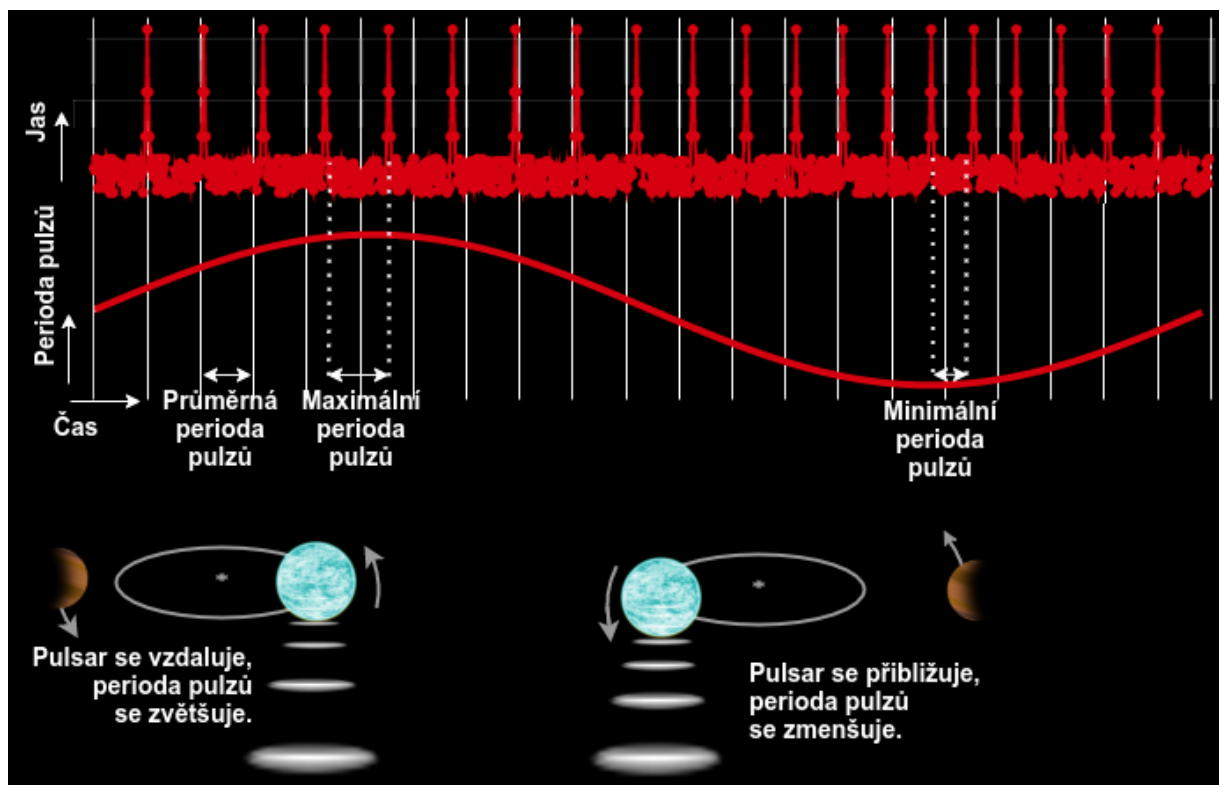
Pulsary jsou rychle rotující neutronové hvězdy – pozůstatky po zhrouceném jádře hmotných hvězd. Během procesu hroucení hvězda zmenšuje svůj poloměr a pro zachování momentu hybnosti zvyšuje svou rotační rychlost. [22]

<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].

Vedle toho, pulsar podél své magnetické osy emituje silný elektromagnetický paprsek. V případě, kdy je magnetická osa pulsaru natočena k Zemi (paprsek směřuje k Zemi), je možno zpozorovat náhlý nárůst jasu hvězdy. Tyto pulzy je pak možné sledovat ve velice pravidelných intervalech (často řádově jednotky milisekund až sekund). [22]

Pokud je v těchto pulzech zpozorována nepravidelnost, může to znamenat, že pulsar mění svou vzdálenost od Země. Jinými slovy, stejně jako v případě metody radiálních rychlostí, pulsar společně s dalším tělesem (např. planetou) obíhá kolem společného těžiště. [22]

Nutno podotknout, že stěžejní veličinou není rychlost pulsaru tak, jako u metody radiálních rychlostí. Nepravidelnosti pulzů jsou způsobeny pouze polohou pulsaru, nikoliv jeho rychlostí. Pulzy (elektromagnetické záření) se ve vakuu pohybují vždy konstantní rychlostí bez ohledu na rychlost pulsaru samotného (rychlosti se nesčítají).



Obrázek 23: Ilustrace pulsaru s obíhající planetou <sup>1</sup>

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io> a GIMP.



## 1.7 Časování tranzitů

V kapitole 1.1 byla popsána tranzitní metoda zkoumající pravidelné poklesy v jasů hvězdy. Někdy ale tyto poklesy nemusí být zcela pravidelné a v jejich periodách mohou být mírné odchylky (TTV – transit time variation). Ty mohou být způsobeny další exoplanetou v systému, která může nebo nemusí sama tranzitovat přes hvězdu, přesto však stále gravitačně ovlivňuje zbytek soustavy. [1]

Situaci lze chápat jako problém n těles, případně ji zjednodušit na problém 3 těles za předpokladu, že v soustavě jsou pouze 3 nezanedbatelně hmotná tělesa:

- Hvězda,
- Tranzitující planeta,
- Planeta způsobující nepravidelnosti.

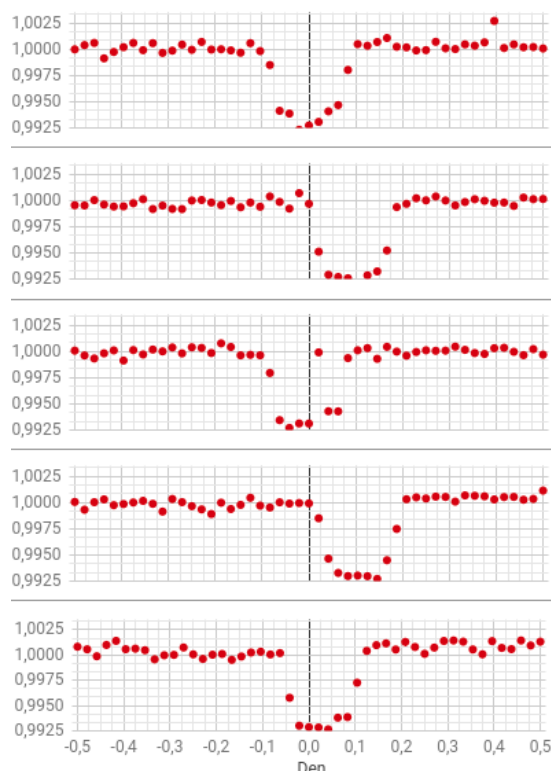
Nechť  $p = \{a, e, i, \Omega, \omega, M, m\}$  je uspořádaná množina obsahující elementy dráhy exoplanety (velkou poloosu, excentricitu, inklinaci, délku vzestupného úhlu, argument šířky pericentra, střední anomálii) a hmotnost exoplanety. Vzhledem k tranzitující planetě budeme tuto množinu označovat jako  $p_t$  a vzhledem k planetě způsobující nepravidelnosti jako  $p_p$ . Další důležitou veličinou je hmotnost hvězdy  $M_s$ . Z těchto údajů jsme schopni spočítat časy tranzitů  $t_{com}$  tranzitující planety. Jestliže máme k dispozici i skutečné naměřené časy tranzitů s nepravidelnostmi  $t_{obs}$ , lze problém řešit jako hledání maxima funkce  $F[t_{com}(p_t, p_p, M_s), t_{obs}]$  definované jako [1]:

$$F = \left[ \sum_{i=1}^N (t_{obs,i} - t_{com,i})^2 \right]^{-1} \quad (9)$$

Vzorec 9: Funkce vyjadřující přesnost modelu vůči realitě při počítání TTV

Jinými slovy, hledáme, jaké parametry planet a hvězdy povedou k nejmenšímu rozdílu mezi napozorovanými a vypočítanými časy tranzitů planety. Čím vyšší je výsledek této

<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].



Obrázek 24: Nepravidelnosti v tranzitech planety Kepler-46b <sup>1</sup>

funkce, o tím přesnější řešení se jedná. Optimální řešení by znamenalo, že jsme zvolili  $p_t$ ,  $p_p$  a  $M_s$  naprosto shodné s realitou a platilo by, že  $F \rightarrow \infty$ .

Kvůli velkému množství vícerozměrných parametrů je nutné pro tento problém použít stochastický optimalizační algoritmus, jenž sice nezajistí přesné analytické řešení, avšak i přesto umožní nalézt řešení velice blízké tomu optimálnímu. Jedním z takových algoritmů je genetický algoritmus blíže popsany v kapitole 3.1. [1]

### 1.7.1 Aplikace genetického algoritmu

Jedince definujeme jako  $x = \{p_t, p_p, M_s\}$ , přičemž první generaci  $K$  jedinců vygenerujeme náhodně. Fitness funkci určující kvalitu jedinců ponecháme jako  $F$  (vzorec 9). S využitím diferenciálních rovnic známých z problému 3 těles:

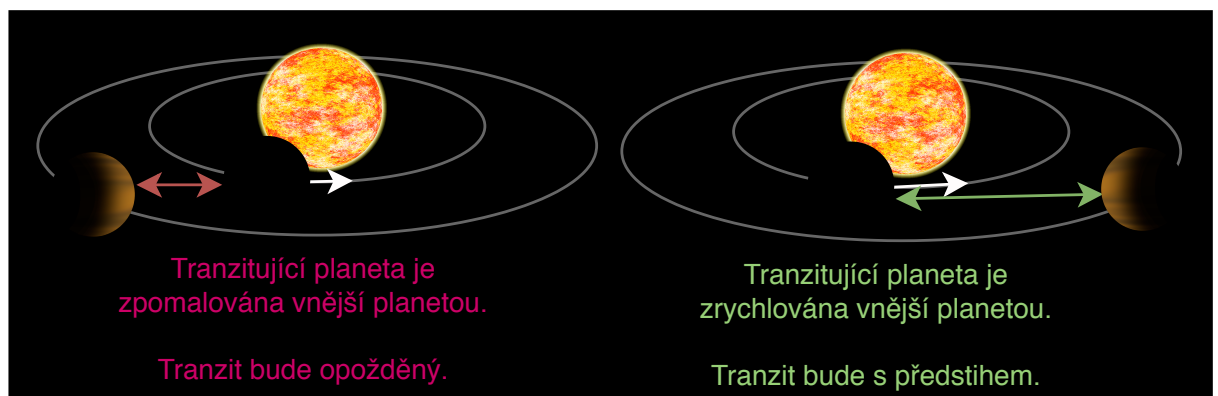
$$\ddot{r}_i = -Gm_j \frac{r_j - r_i}{|r_j - r_i|^3} - Gm_k \frac{r_k - r_i}{|r_k - r_i|^3} \quad (10)$$

Vzorec 10: Rovnice problému 3 těles

$r_i$ = polohový vektor i-tého tělesa	$m_i$ = hmotnost i-tého tělesa	$G$ = gravitační konstanta
$i$ = aktuální těleso	$j, k$ = další 2 tělesa	

Kde  $r_i = (x_i, y_i, z_i)$ , lze pomocí fitness funkce vypočítat kvalitu každého z jedinců v populaci. Následně náhodně, avšak v souladu s přirozeným výběrem s přihlédnutím na fitness vybereme dvojici jedinců, na něž bude aplikována operace křížení, jejímž výsledkem budou 2 potomci. [1]

Tímto způsobem se pokračuje dále pro získání celé populace potomků, kteří navíc mohou mít nenulovou pravděpodobnost mutace. Z populace rodičů a potomků vhodně vybereme  $K$  jedinců, kteří se stanou 2. generací. Běh algoritmu bude pokračovat dokud nejlepší jedinec poslední generace nebude představovat dostatečně dobré řešení TTV.



Obrázek 25: Princip časování tranzitů <sup>1</sup>

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io> a GIMP.

## 2 UMĚLÉ NEURONOVÉ SÍTĚ

Jednou z oblastí patřících do oboru umělé inteligence jsou umělé neuronové sítě, jež se inspirovaly biologickými neuronovými sítěmi. Jejich použití je vhodné na problémy, u kterých neexistuje přesný matematický popis řešení nebo sice existuje, ale jeho realizace by byla příliš složitá. Hlavními oblastmi, kde se neuronové sítě využívají, jsou např.:

- **Predikce** (předpověď počasí, vývoj cen akcií na burze, ...),
- **Rozpoznávání vzorů** (rozpoznávání ručně psaného textu, detekce tranzitu planety ve světelné křivce hvězdy, ...),
- **Analýza nebo transformace signálů** (odstranění šumu ze signálu, komprese, převod psaného textu na mluvený signál, ...),
- **Řízení v dynamicky se měnících podmínkách** (autopilot, ...). [3]

### 2.1 Neuron

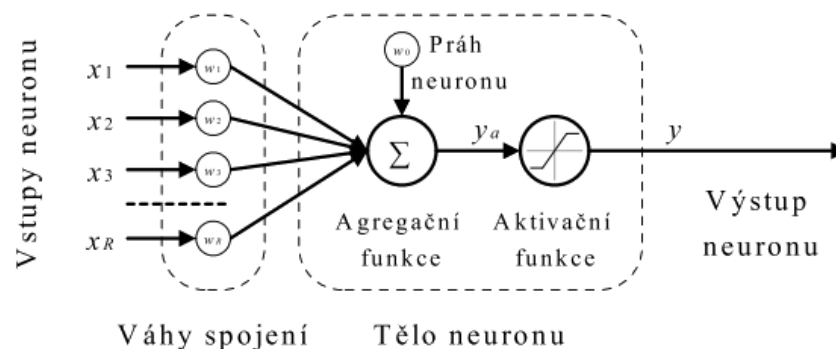
Základním stavebním prvkem umělé neuronové sítě je umělý neuron, který reprezentuje zjednodušenou biologickou nervovou buňku. Nejpoužívanějším typem umělého neuronu je tzv. formální neuron. Ten se skládá z následujících částí:

Komponenta	Popis	Příklad
Vstupy $x_{1...R}$	Jeden nebo více vstupů.	[0.23, 0.58, -0.15]
Váhy spojení (vstupů) $w_{1...R}$	Každý vstup má svou váhu.	[0.01, 0.56, 0.12]
Práh neuronu $w_0$	Přičte se ke vstupu do aktivační funkce.	0.17
Agregační funkce $\sum$	Vstupy a váhy transformuje na jednu hodnotu – potenciál $y_a$ .	$y_a = w_0 + \sum_{n=1}^R x_n w_n$
Aktivační funkce	Převede hodnotu vstupního potenciálu $y_a$ na výstup neuronu $y$ .	$y = y_a$ $y = \tanh y_a$

Tabulka 5: Komponenty formálního neuronu <sup>1</sup>

Jednotlivé neurony se umísťují do vrstev a vrstvy pak tvoří neuronovou síť. Způsob zapojení neuronů, počet neuronů, počet vrstev a stejně tak volba agregační a aktivační funkce závisí na typu problému, pro který má být neuronová síť použita.

<sup>1</sup> Zdroj: [3]



Obrázek 26: Model formálního neuronu <sup>1</sup>

### 2.1.1 Vstupy neuronu

Každý neuron může mít jeden nebo více vstupů. I když vstupy nemusí být nutně číselné, v této práci to bude předpokládáno. Vstupy mohou být dvojího druhu:

- Výstup jiného neuronu,
- Vstup z vnějšího prostředí (např. od uživatele).

Všechny vstupy neuronu je možno reprezentovat vektorem  $X = [x_1, x_2, \dots, x_R]$ . Hodnoty v  $X$  pak mohou být v:

- **Kvantitativní** formě, kdy vstup vyjadřuje *ano* nebo *ne* a nabývá tak binárních hodnot (např. 0 nebo 1),
- **Kvalitativní** formě, kdy vstup vyjadřuje konkrétní hodnotu nějaké veličiny, často normalizované na interval  $[0; 1]$  nebo  $[-1; 1]$ .

### 2.1.2 Váhy spojení

Každý vstup neuronu  $x_i$  je ohodnocen číselnou vahou  $w_i$ . Čím vyšší váha vstupu je, tím citlivěji bude výstup neuronu reagovat, pokud se daný vstup změní (tím „důležitější“ vstup bude). Váhy jsou z počátku většinou pouze náhodné hodnoty a teprv během procesu učení neuronové sítě dochází k jejich úpravě tak, aby síť byla schopna řešit předložený problém co nejlépe. Vztah vstupu a jeho váhy se obecně označuje operátorem konfluence. [3]

$$z_i = x_i \oplus w_i \quad (11)$$

Vzorec 11: Obecný vztah vstupu neuronu a jeho váhy

Formální neuron pak využívá lineární hodnotu tohoto operátoru, který je tak možné nahradit prostým součinem. [3]

<sup>1</sup> Převzato z: [3].

$$z_i = x_i w_i \quad (12)$$

Vzorec 12: Lineární vztah vstupu neuronu a jeho váhy

### 2.1.3 Práh neuronu

Prahem umělého neuronu  $w_0$  se rozumí bariéra vstupu do neuronu z vnějšího okolí (nikoliv z jiného neuronu). Vedle váhy je to další parametr, který se mění při procesu učení sítě. [3]

### 2.1.4 Agregáčn  funkce

Protože neuron m že m t v c vstup , ale v dy pouze jedin  v stup, je t eba v echny vstupy n jak m zp sobem slou it do jedn  hodnoty. Pr v  toho se sna ı doc lit agregáčn  funkce neuronu. V p ıpad  form ln ho neuronu se nej ast ji pou ıv  prost  suma sou in  vstup  s v hami. V stupem agregáčn  funkce je potenci l  $y_a$ . [3]

$$y_a = \sum_{i=0}^R z_i = \sum_{i=0}^R w_i \oplus x_i = \sum_{i=0}^R w_i x_i \quad (13)$$

Vzorec 13: Agregáčn  funkce neuronu

### 2.1.5 Aktiva n  funkce

Je t e n   je potenci l  $y_a$  p ıveden na v stup neuronu, je na n   aplikov na aktiva n  funkce. Tato funkce je z visl  na typu ře en ho probl mu a stejn  tak i na poloze neuronu v s ti. Je b  n ,  e pro v stupn  neurony se pou ıv j  jin  aktiva n  funkce, n   pro neurony ve skryt ch vrstv ch. Funkce mohou b t line rn , neline rn , spojite i diskretn . [3]

$$y = \tanh y_a \quad (14)$$

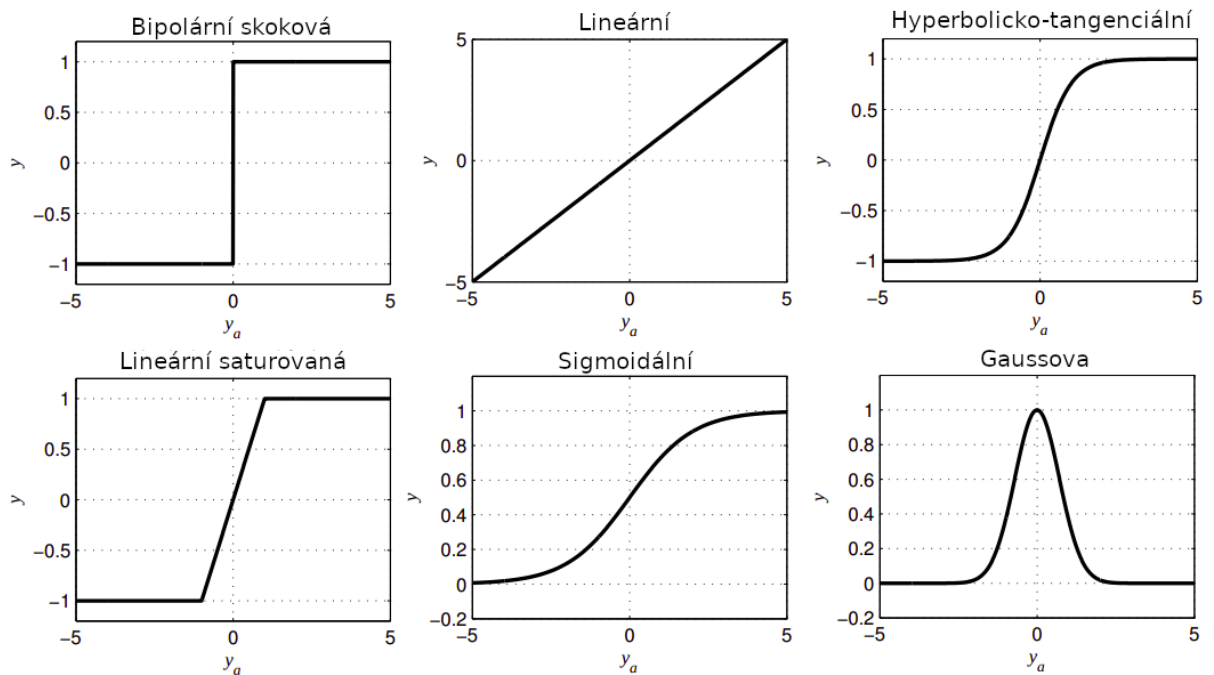
Vzorec 14: Hyperbolicko-tangenci ln  aktiva n  funkce

$$y = \frac{e^x}{1 + e^x} \quad (15)$$

Vzorec 15: Sigmoid ln  aktiva n  funkce

$$y = e^{-y_a^2} \quad (16)$$

Vzorec 16: Gaussova aktiva n  funkce



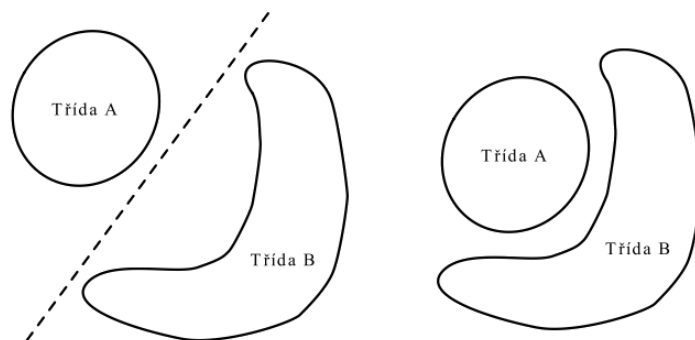
Obrázek 27: Příklady aktivačních funkcí neuronu <sup>1</sup>

## 2.2 Typy neuronových sítí

V této kapitole budou zmíněny některé typy neuronových sítí. Uvedený výčet však v žádném případě není kompletní. Vedle zmíněných typů existují např. rekurentní samoorganizující se neuronové sítě.

### 2.2.1 Jednoduchý perceptron

Jednoduchý perceptron je neuronová síť tvořená jediným neuronem. Jedná se o neuronovou síť s lineárně váženou agregační funkcí, učení s učitelem a skalárním výstupem nabývajícím binárních (1, 0) nebo bipolárních (1, -1) hodnot. [3]



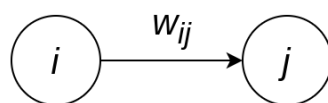
Obrázek 28: Lineárně separovatelná (vlevo) a neseparovatelná (vpravo) úloha <sup>1</sup>

<sup>1</sup> Převzato z: [3].

Tento typ neuronové sítě lze použít pouze pro řešení úloh, které jsou lineárně separovatelné. Pro složitější úlohy je třeba využít vícevrstvý perceptron. [3]

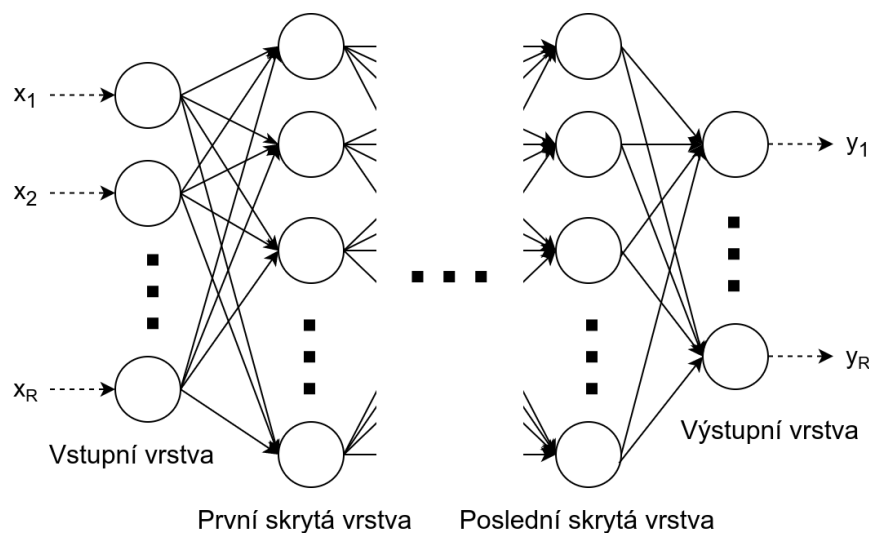
### 2.2.2 Dopředná vícevrstvá umělá neuronová síť

Dopředná vícevrstvá umělá neuronová síť (dále jen FFNN, někdy také vícevrstvý perceptron) je díky své univerzálnosti jedním z nejpoužívanějších typů sítí. Skládá se z neuronů, které jsou umístěny do jednotlivých vrstev. Existuje zde minimálně vrstva vstupní a vrstva výstupní. Dle povahy řešeného problému zde ale může být i libovolný počet dalších, tzv. skrytých vrstev. [3]



Obrázek 29: Zapojení neuronů <sup>1</sup>

Spojení existují pouze mezi neurony sousedních vrstev. Ani neurony stejné vrstvy, ani neurony nesousedících vrstev nejsou přímo propojeny. Orientace těchto spojů je navíc směrem pouze od vstupů k výstupu sítě (proto dopředná) – výstupy neuronů v předešlé vrstvě jsou vstupem do všech neuronů v následující vrstvě. [3]



Obrázek 30: Architektura dopředné vícevrstvé umělé neuronové sítě <sup>1</sup>

Pro vytváření architektury neuronové sítě neexistuje analytický postup, který by vedl k nejefektivnějšímu návrhu. Při řešení úlohy je třeba buď inspirovat se již nějakou existující neuronovou sítí, s jejíž pomocí někdo podobný problém řešil v minulosti, nebo postupovat

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

experimentálně a „zkoušet“, jaká architektura povede k nejmenší chybě. V tomto případě je možno postupovat dvěma způsoby. Buď začneme s triviální sítí, do které postupně budeme přidávat nové neurony a vrstvy, až dokud bude úspěšnost sítě při řešení úkolu stoupat, nebo naopak začneme s komplexní neuronovou sítí, ze které budeme neurony a vrstvy postupně odebírat. [3]

### 2.2.3 Konvoluční neuronová síť

Zpracování vícedimenzionálních dat o velkých rozměrech by s použitím klasické FFNN bylo velice problematické. Uvažujme např. rozpoznávání psaných číslic z obrázku. Číslice se v obrázku může nacházet na různých místech, může být různě velká, natočená či barevná a také způsob psaní se u každého člověka může lišit.

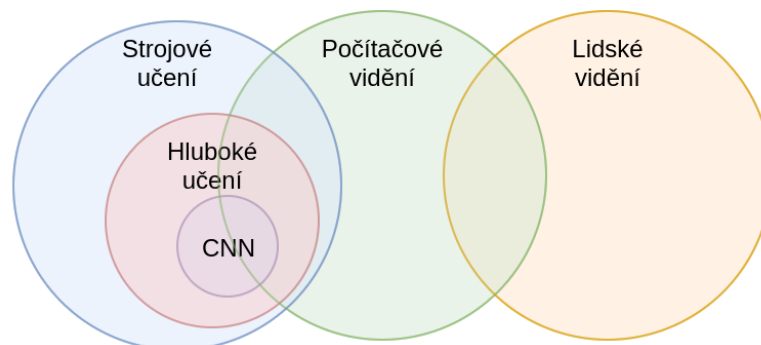


Obrázek 31: Ručně psané číslice 9. <sup>1</sup>

Pro člověka je jednoduché rozpoznat, že všechny útvary na obrázku 31 jsou číslice 9 i přesto, že každá vypadá jinak. Sestrojit algoritmus, který by to dokázal také, už ale triviální úkol rozhodně není. Obecně, získáváním informací z obrázku se zabývá obor počítačového vidění. [5]

Zásadní myšlenkou počítačového vidění je nesnažit se rozpoznávat celý objekt, ale nejdříve extrahovat fragmenty (vlastnosti), ze kterých se daný objekt skládá. U číslice 9 by těmito vlastnostmi mohly být např. elipsa a z ní vycházející křivka či úsečka. K extrakci vlastností z obrázku lze přistupovat dvěma způsoby založenými na:

- Manuálním inženýrství (např. histogram orientovaných gradientů),
- Strojovém učení (např. konvoluční neuronová síť). [5]



Obrázek 32: Vztah mezi počítačovým viděním, strojovým učení a konvoluční sítí <sup>2</sup>

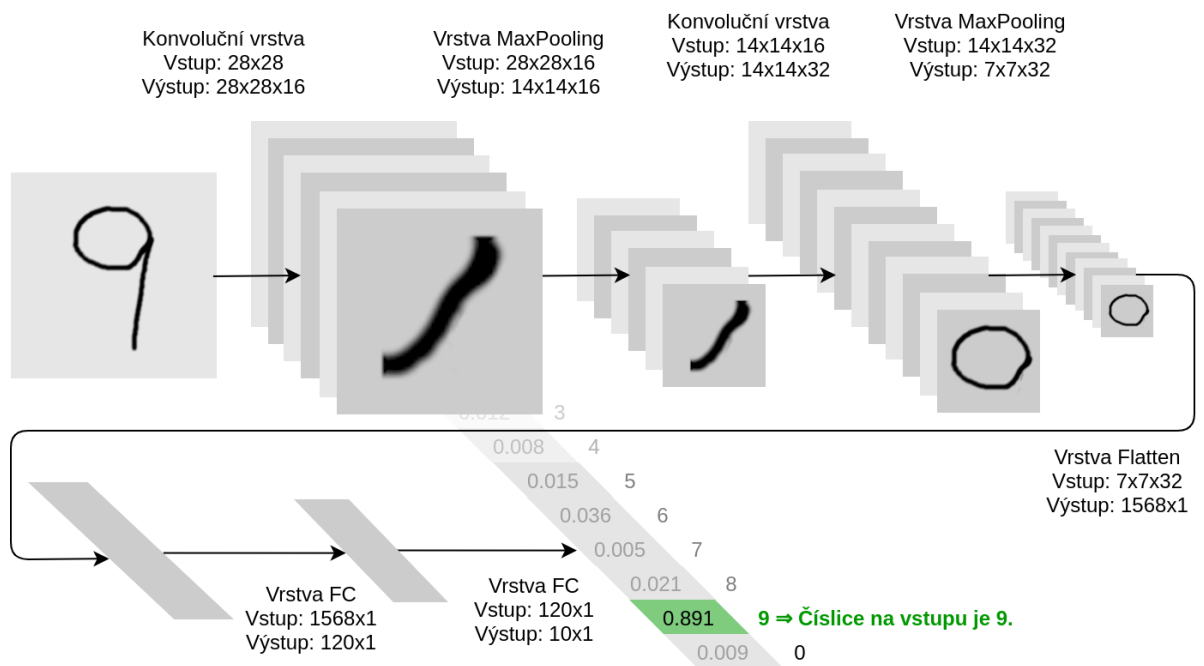
<sup>1</sup> Převzato z: MNIST dataset.

<sup>2</sup> Vytvořeno autorem v <https://www.draw.io>.



Zde bude probrána pouze konvoluční neuronová síť (CNN). Tu lze chápat jako klasickou FFNN, jejíž vstupy jsou ovšem nejdříve modifikovány tak, aby z nich byly vyextrahovány výše zmíněné vlastnosti. Tato extrakce je prováděna pomocí konvoluční vrstvy, jejíž způsob fungování je detailněji popsán v kapitole 2.3.2.

Dále je třeba brát v úvahu velikost obrázku. Full HD obrázek (1 920 \* 1 080) se skládá z více jak 2 milionů pixelů. V případě RGB obrázku se množství informací zvýší až na 6 milionů (3 složky pro každý pixel). Posílat takové množství hodnot do FFNN, která se skládá z plně propojených vrstev, kde jsou neurony v sousedních vrstvách propojeny způsobem „každý z každým“, by bylo problematické. Z tohoto důvodu existuje tzv. pooling vrstva popsaná v kapitole 2.3.3, která umožňuje zmenšit velikost vstupu a přitom zachovat vzory charakteristické pro vstup.



Obrázek 33: Příklad konvoluční sítě pro rozpoznávání ručně psaných číslic <sup>1</sup>

Nutno podotknout, že použití CNN se neomezuje pouze na dvourozměrné obrázky. Stejně vhodné jsou i na rozpoznávání vzorů v 1D útvech (např. světelná křivka hvězdy nebo jiné druhy časových řad) nebo i ve vícerozměrných strukturách. [5]

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io> a GIMP.

## 2.3 Typy vrstev neuronových sítí

### 2.3.1 Plně propojená vrstva

Plně propojená vrstva (FC, Fully connected nebo Dense layer) je vrstva, pro jejíž každý neuron platí, že na jeho vstup jsou přivedeny výstupy všech neuronů v předchozí vrstvě. Neurony sousedních vrstev FC jsou tak mezi sebou propojeny stylem „každý s každým“.

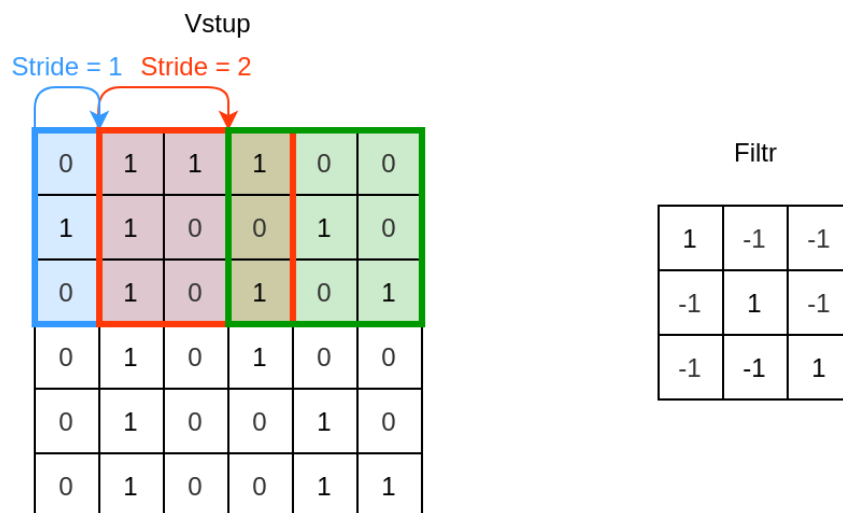
Parametr	Popis
Počet neuronů	Je určen složitostí a typem řešeného problému a určuje velikost výstupu vrstvy.

Tabulka 6: Parametry plně propojené vrstvy <sup>1</sup>

Počet neuronů se volí dle řešeného problému. Pokud bude počet neuronů příliš nízký, neuronová síť nemusí mít kapacitu pro to, aby byla schopna se naučit řešit daný problém. Pokud bude počet neuronů příliš vysoký, síť se až příliš dobře naučí řešit problém na trénovací množině a nebude schopna generalizace (snadno se tzv. přetrénuje).

### 2.3.2 Konvoluční vrstva

Konvoluční vrstva sestává z filtrů, které jsou aplikovány na vstup a provádí operaci „konvoluce“. Filtrem lze chápat vzor, jenž je vyhledáván ve vstupu a který zastupuje nějakou vlastnost vstupního objektu (např. obrázku).



Obrázek 34: Postupné posouvání konvolučního filtru přes vstup <sup>2</sup>

<sup>1</sup> Zdroj: [21]

<sup>2</sup> Vytvořeno autorem v <https://www.draw.io>.

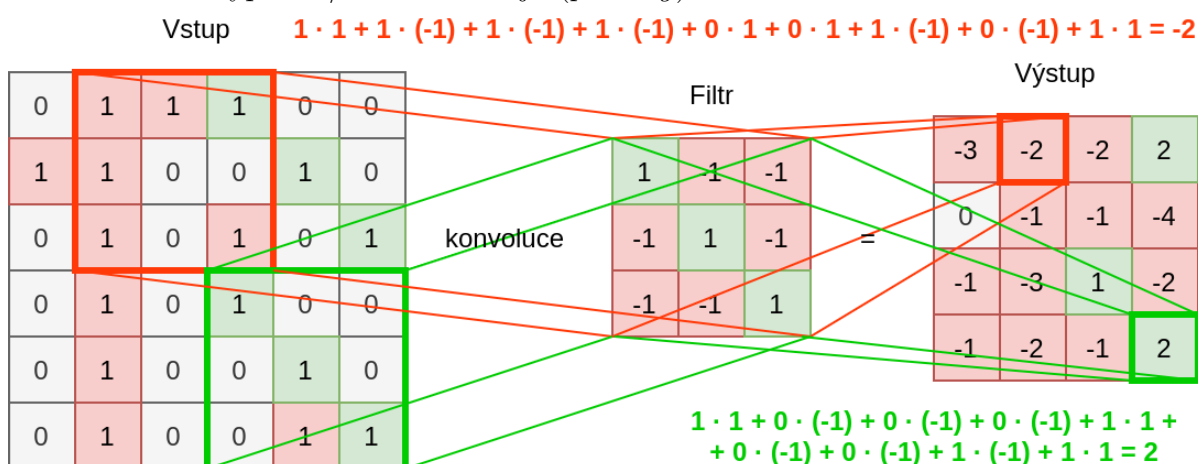
Pro výřez ze vstupu a filtr v konvoluční vrstvě se hodnoty na stejných pozicích ve výřezu i filtru vynásobí a tyto součiny se následně sečtou.

$$y = \sum_{i=1}^{sy} \sum_{j=1}^{sx} v_{ij} f_{ij} \quad (17)$$

Vzorec 17: Provádění konvoluce pro výřez vstupu a filtr

$y$  = výstup |  $sx, sy$  = horizontální/vertikální velikost filtru |  $v$  = výřez vstupu |  $f$  = filtr

Výřez se postupně posouvá (*stride*) v rámci vstupu a pro každou kombinaci výřezu a daného filtru se výsledek uloží do výstupního tenzoru na odpovídající místo vzhledem ke vstupu. Výsledkem je tak tenzor (*feature map*) stejně velký nebo menší, než vstup – v závislosti na vyplnění/oříznutí okrajů (*padding*).



Obrázek 35: Příklad aplikace konvolučního filtru na vstup <sup>1</sup>

Na obrázku 35 je znázorněno rozpoznávání diagonály. Ve výstupu jsou v oblastech, kde jsou ve vstupu diagonály (v pravém horním i dolním rohu), vyšší číselné hodnoty než tam, kde ve vstupu diagonály nejsou. Celý tento postup se opakuje pro všechny filtry, jimiž konvoluční vrstva disponuje. Výstupem vrstvy je tak jedna nebo více *feature maps*. [5]

Parametr	Popis
Počet filtrů	Čím více filtrů, tím více vlastností ve vstupu lze vyhledávat.
Velikost filtrů (kernel)	Čím větší filtr, tím složitější vzory lze ve vstupu vyhledávat. Filtr může mít v každém svém rozměru jinou velikost (v případě 2D to nemusí být čtverec, ale i obdélník).
Posunutí filtrů (stride)	Posunutí filtru v každé fázi konvoluce.
Okraj (padding)	Způsob zpracování okrajů vstupu, které se ve výstupu buď useknou, nebo vyplní nějakou hodnotou.

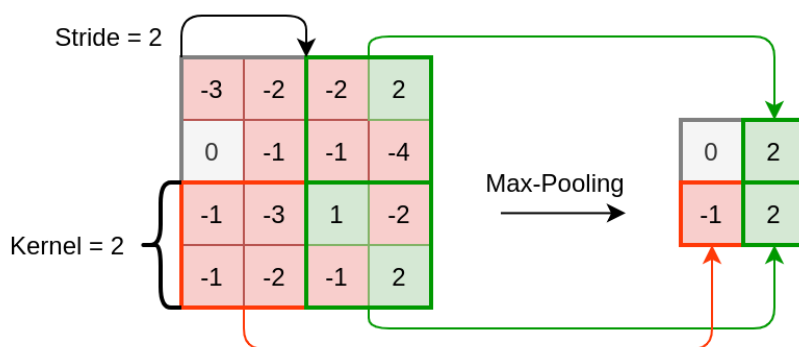
Tabulka 7: Parametry konvoluční vrstvy <sup>2</sup>

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

<sup>2</sup> Zdroj: [21]

### 2.3.3 Vrstva Max-Pooling

Vrstva Max-Pooling si klade za cíl zmenšit velikost vstupu a zároveň zachovat co nejvíce důležitých informací, které se v něm nachází. Podobně jako v konvoluční vrstvě i zde se posouvá okno o určité velikosti (kernel) o určitý počet polí (stride). Na výstup se však dostane maximální hodnota, která je v aktuálním výřezu vstupu. [5, 21]



Obrázek 36: Příklad operace Max-Pooling <sup>1</sup>

Na obrázku 36 je ukázána aplikace operace Max-Pooling na vstup, který byl výstupem v minulé kapitole – *feature map* ukazující umístění diagonál v původním obrázku. Jak je vidět, velikost struktury se zmenšila na čtvrtinu, a přesto v ní zůstaly všechny důležité informace. V pravém dolním i horním rohu jsou stále vysoké číselné hodnoty indikující přítomnost diagonály.

Parametr	Popis
Velikost okna (kernel)	Velikost okna, ze kterého se vybírá maximum.
Stride okna (stride)	Posunutí okna v každé fázi operace Max-Pooling.
Okraj (padding)	Způsob zpracování okrajů vstupu, které se ve výstupu buď useknou nebo vyplní nějakou hodnotou.

Tabulka 8: Parametry vrstvy Max-Pooling <sup>2</sup>

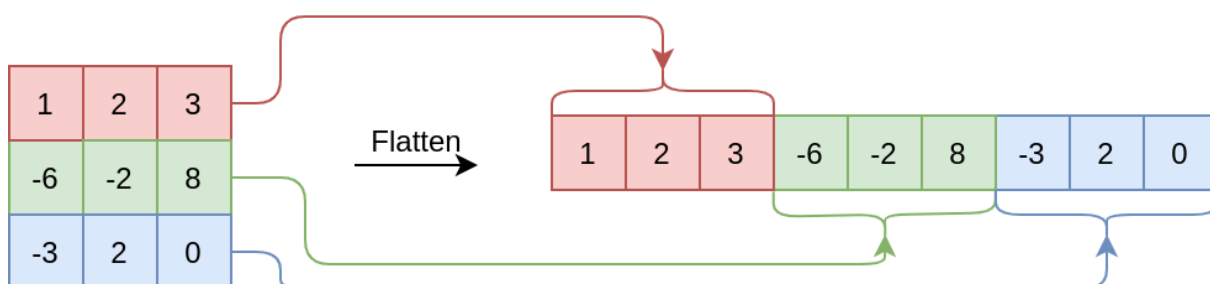
Obdobně k vrstvě Max-Pooling existuje např. i Min-Pooling nebo Average-Pooling. Jejich fungování je stejné s tím rozdílem, že na výstup předávají místo maximální hodnoty tu minimální, resp. průměrnou. Jejich použití je opět závislé na typu řešeného problému (např. zda vstupní obrázek má světlé pozadí a tmavé popředí či naopak). [21]

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

<sup>2</sup> Zdroj: [21]

### 2.3.4 Vrstva Flatten

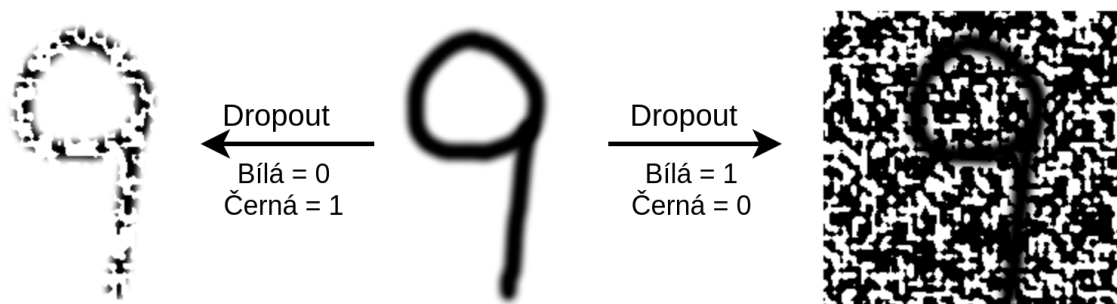
Konvoluční vrstvy umí pracovat s vícerozměrnými vstupy. Vrstvy některých jiných typů však často očekávají pouze jednorozměrný vektor vstupních hodnot. Je proto nutné nějakým způsobem převést n-rozměrný tenzor na vektor. Právě k tomu slouží vrstva Flatten. [21]



Obrázek 37: Příklad fungování vrstvy Flatten <sup>1</sup>

### 2.3.5 Vrstva Dropout

Pro snížení pravděpodobnosti, že se neuronová síť přetrénuje, je možné využít vrstvu Dropout. Ta při procesu trénování zahazuje vstupy (na výstup místo nich posílá hodnotu 0). Tím v trénovací množině vzniká náhodný šum, což způsobí, že síť nebude trénovat pouze na stále stejných vstupech, ale trénovací množina bude vždy nepatrně odlišná – různorodější. [21]



Obrázek 38: Příklad fungování vrstvy Dropout <sup>1</sup>

Parametr	Popis
Rate	Frekvence zahazování vstupů.

Tabulka 9: Parametry vrstvy Dropout <sup>2</sup>

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io> a GIMP.

<sup>2</sup> Zdroj: [21]

## 2.4 Učení umělé neuronové sítě

Vytvořená neuronová síť dle předchozích kapitol nebude umět řešit žádný problém – je to pouze množina neuronů a dalších parametrů poskládaných k sobě. Je potřeba ji daný problém nejdříve naučit řešit. Učení je proces, při kterém síť upravuje nastavitelné parametry (např. váhy vstupů neuronů) za účelem zvýšení úspěšnosti řešení předkládané úlohy. Učení může být:

- **S učitelem** – Síti je předložen vstup z trénovací množiny a ta pro tento vstup stanoví odezvu (výstup), který je porovnán s požadovaným výstupem. Na základě chyby mezi skutečným a požadovaným výstupem pak síť za využití tzv. učícího algoritmu upraví váhy mezi neurony tak, aby minimalizovala chybovou funkci.
- **Bez učitele** – Neuronová síť na základě schopnosti rozeznávat ve svých vstupech podobné vlastnosti a podle těchto vlastností třídit vstupy dokáže řešit určité problémy i bez znalosti požadovaných výstupů. [3]

### 2.4.1 Algoritmus zpětného šíření chyby

Algoritmus zpětného šíření chyby je učící algoritmus určený pro FFNN s diferencovatelnými spojitými aktivačními funkcemi. Neuronové síti je předložen vzor, pro který síť stanoví odezvu. Chyba pro jeden vzor je definována jako [3]:

$$E = \sum_{i=1}^Q (t_i - y_i)^2 = \sum_{i=1}^Q e_i^2 \quad (18)$$

Vzorec 18: Výpočet chyby pro jeden vzor u algoritmu zpětného šíření chyby

$$\begin{array}{l} E = \text{chyba jednoho vzoru} \\ Q = \text{počet výstupů NN} \end{array} \left| \begin{array}{l} y_i = \text{Skutečná hodnota } i\text{-tého výstupu} \\ t_i = \text{Očekávaná hodnota } i\text{-tého výstupu} \end{array} \right. \begin{array}{l} e_i = \text{Chyba } i\text{-tého výstupu} \end{array}$$

Následně je ke každé váze připočten přírůstek  $\Delta w_{ij}^k$  (při online učení – rychlejší) nebo se přírůstek kumuluje a přičte se až na konci epochy (při offline učení – stabilnější). [3]

$$\Delta w_{ij}^k = \frac{\partial E}{\partial w_{ij}^k} \implies \Delta w_{ij}^k = \alpha \delta_j^k y_i^{k-1} \quad (19)$$

Vzorec 19: Přírůstek váhy u algoritmu zpětného šíření chyby

$$\begin{array}{l} w_{ij}^k = \text{Váha spoje mezi } i\text{-tým a } j\text{-tým neuronem v } k\text{-té vrstvě} \\ \delta_j^k = \text{lokální gradient neuronu aktualizované váhy} \\ y_j^{k-1} = \text{výstup } j\text{-tého neuronu v } k-1. \text{ vrstvě} \end{array} \left| \begin{array}{l} \alpha = \text{rychlost učení} \\ E = \text{chyba jednoho vzoru} \end{array} \right.$$

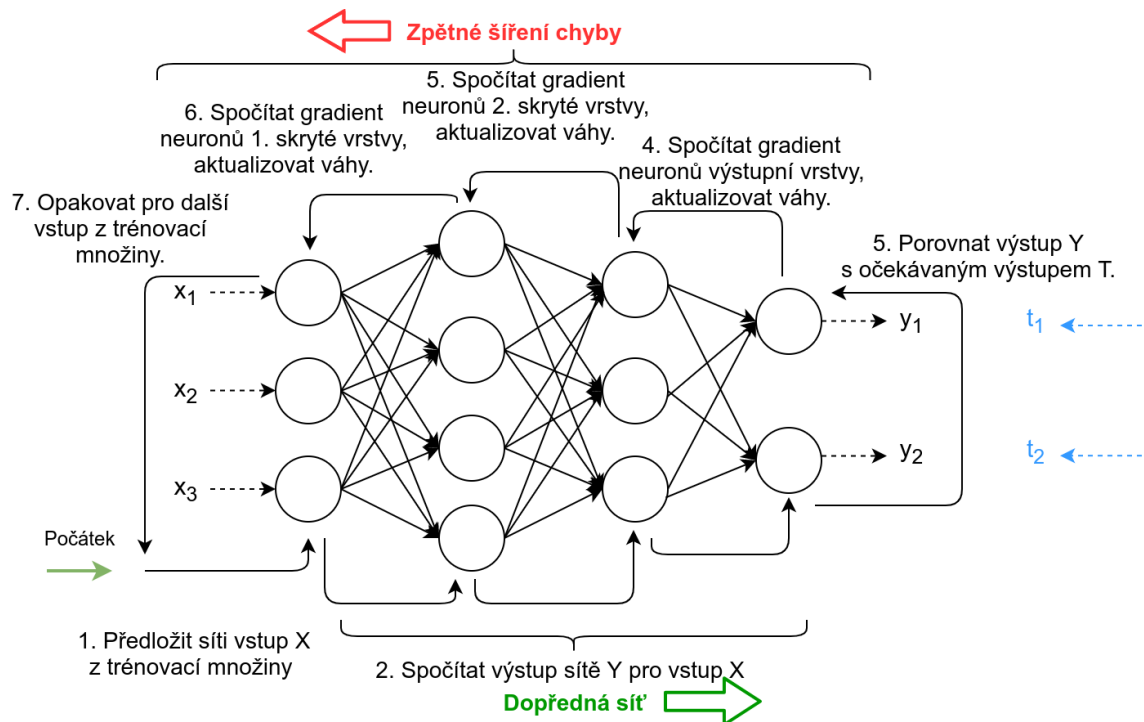
Přičemž lokální gradient neuronu je vypočítán dle vzorce 20. Tento postup je opakován pro všechny vzory v trénovací množině – této iteraci se říká *epocha*. Učení sestává z jedné nebo více epoch a může skončit:

- Po pevném počtu epoch,
- Po neurčitým počtu epoch, až chyba učení na klesne na určitou hodnotu. [3]

$$\delta_j^k = \begin{cases} \phi^k(y_{aj}^k) \sum_l \delta_l^{k+1} w_{jl}^{k+1}, & \text{pro neurony ve skrytých vrstvách} \\ e_j \phi^k(y_{aj}^k), & \text{pro neurony ve výstupní vrstvě} \end{cases} \quad (20)$$

Vzorec 20: Výpočet lokálního gradientu neuronu

$$\begin{array}{l|l} \delta_l^{k+1} = \text{lokální gradient } l\text{-tého neuronu } k+1. \text{ vrstvy} & e_j = \text{chyba výstupu } j\text{-tého neuronu} \\ y_{aj}^k = \text{vstupní potenciál } j\text{-tého neuronu } k\text{-té vrstvy} & \phi^k = \text{derivace aktivační funkce } k\text{-té vrstvy} \\ w_{ij}^k = \text{váha spoje mezi } i\text{-tým a } j\text{-tým neuronem } k\text{-té vrstvy} & \end{array}$$



Obrázek 39: Příklad algoritmu zpětného šíření chyby <sup>1</sup>

Počáteční hodnoty vah spojují mezi neurony mohou být nastaveny např. náhodně za použití rovnoměrného nebo normálního rozdělení s nulovou střední hodnotou. Rychlost učení  $\alpha$  určuje rychlost, s jakou se váhy spojují mezi neurony upravují dle chyby při trénování. Obvykle nabývá hodnot jako 0,1, 0,01, 0,001, apod. [3]

Čím vyšší rychlost učení je, tím rychleji bude síť natrénovaná, ale tím pravděpodobněji trénování skončí v nějakém lokálním minimu chybové funkce a nebude dosaženo tak dobrého výsledku. [5]

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

### 3 DALŠÍ OBLASTI UMĚLÉ INTELIGENCE

Umělá inteligence (AI) je podstatně rozsáhlejší obor a neobsahuje pouze umělé neuronové sítě. Další podoblasti však nejsou natolik důležité v kontextu této práce, a proto zde budou popsány pouze okrajově. Ačkoliv se definice v různých zdrojích různí, AI lze popsat jako odvětví informatiky zabývající se automatizací inteligentního chování strojů. [13] Lze ji rozdělit na 2 základní kategorie:

- **Silná umělá inteligence** – hypotetická obecná inteligence stroje umožňující řešit libovolný problém, který dokáže řešit člověk.
- **Slabá umělá inteligence** – skupina algoritmů, technik a postupů umožňujících řešit nějaký konkrétní a specifický problém neřešitelný nebo těžko řešitelný běžnými analytickými metodami, avšak často lehce řešitelný člověkem samotným. [13]

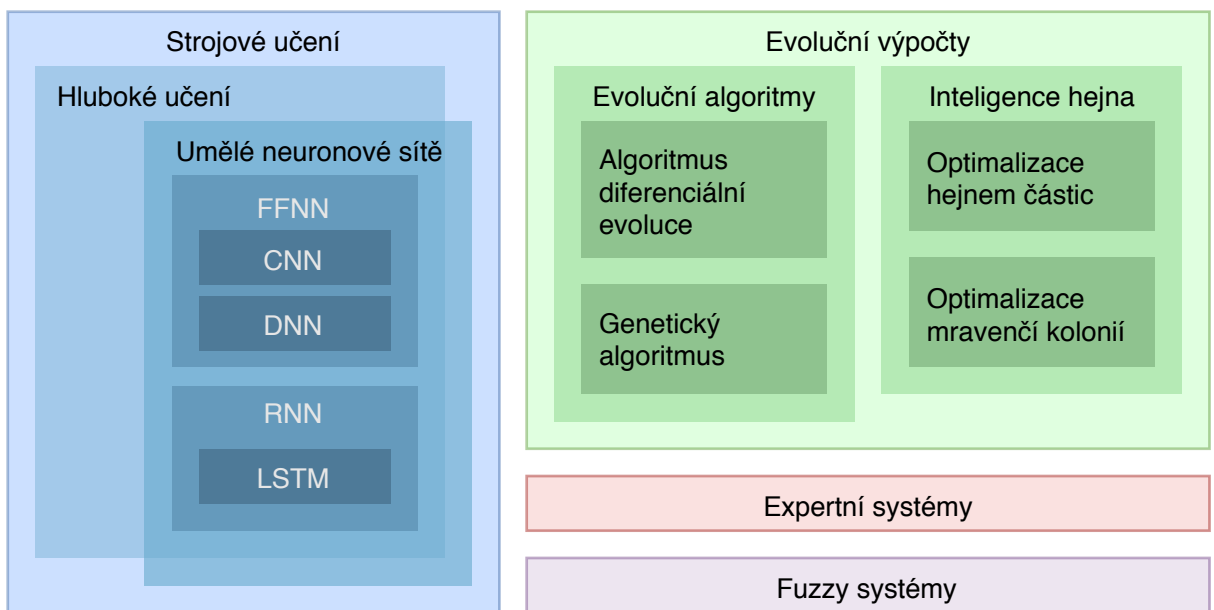
Příčemž slabou umělou inteligenci lze dále dělit např. na:

- **Strojové učení** – část AI umožňující strojům učit se z dat a modifikovat tak své chování bez zásahu člověka. [13]
  - **Hluboké učení** – specifická část strojového učení spolehlující se na vícevrstvou interpretaci řešeného problému. Umožňuje hlubší pochopení dané problematiky a více se tak blíží fungování lidského mozku.
  - **Umělé neuronové sítě** – výpočetní modely vycházející z biologických neuronových sítí. Existují jednoduché nebo i vícevrstvé sítě, dále dopředné, rekurentní, konvoluční, samoroorganizační nebo třeba modulární. Některé z těchto typů sítí byly detailněji popsány v kapitole 2.
- **Evoluční výpočty** - podoblast AI napodobující biologické procesy. Často se jedná o metaheuristické optimalizační techniky, které kromě povahy řešeného problému nepotřebují žádné další informace a řešení (byť ne vždy optimální) najdou samy. [13]
  - **Evoluční algoritmy** – snaží o napodobení evoluce a vychází z pravidla přežití nejsilnějšího (nejpřizpůsobivějšího). Často se zde vyskytují pojmy jako generace, přirozený výběr, dědičnost nebo křížení. Spadá sem např. genetický algoritmus nebo algoritmus diferenciální evoluce.
  - **Intelligence hejna** – simulace sociální inteligence v nějaké množině převážně jednoduchých a decentralizovaných jedinců. Patří sem např. optimalizace hej-



nem částic vycházející z chování hejna ptáků nebo optimalizace mravenčí kolonií, jež se inspirovala mravenci a jejich komunikací pomocí feromonů.

- **Expertní systémy** – systémy řešící nenumernické problémy umožňující poskytovat expertní rady a doporučení v konkrétních situacích. K tomu využívají bázi znalostí, v níž mají uloženy všechny známé informace o problému.
- **Fuzzy systémy** – systémy založené na fuzzy logice. Ta narozdíl od té výrokové nepracuje s binárními hodnotami (pravda, nepravda), ale pravdivost tvrzení lze vyjádřit jakoukoliv hodnotou z intervalu  $\langle 0; 1 \rangle$ . Fuzzy (mlhavá, nejasná) není exaktní a jednoznačná a více se tak blíží lidskému chápání světa. [13]



Obrázek 40: Rozdělení umělé inteligence <sup>1</sup>

### 3.1 Genetický algoritmus

Genetický algoritmus (GA) je metaheuristický postup inspirovaný evolučními pravidly, která je možno vidět v přírodě kolem nás. Obvykle se používá pro řešení optimalizačních problémů, pro něž neexistuje alternativní použitelný analytický postup. [1, 4]

Jako příklad lze uvést Problém obchodního cestujícího (Traveling salesman problem – TSP) patřící mezi NP-těžké úlohy: „*Existuje  $n$  měst propojených přímými cestami. Úkolem je navštívit všechna tato města, vrátit se do města počátečního a přitom urazit co nejkratší*

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

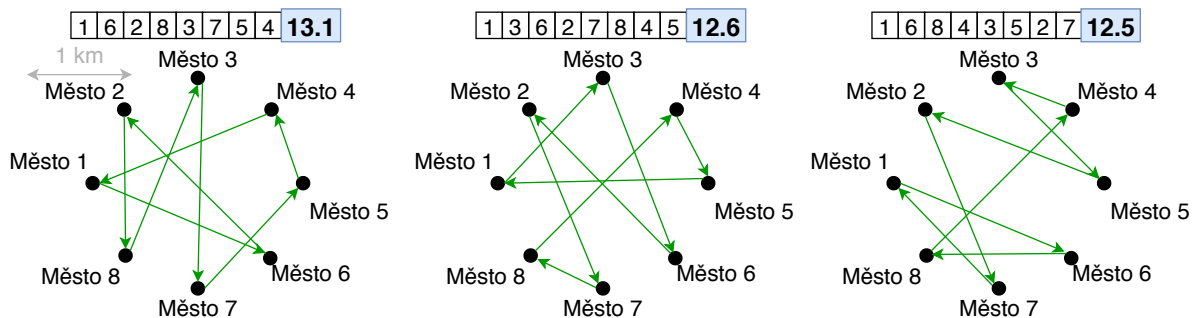
vzdálenost.“ Se vzrůstajícím  $n$  začne počet možných cest rychle narůstat a hledat z nich tu nejkratší exaktními postupy nebo dokonce hrubou silou se tak stane časově neúnosným. [4]

1. Define  $K$  (population size),  $I$  (number of generations),  $M$  (mutation probability) a  $F$  (fitness function).
2. Generate  $K$  random individuals (first generation).
3. Apply  $F$  to all individuals in first generation.
4. Repeat from 1 to  $I$ :
  - 4.1. Repeat from 1 to  $K$ :
    - 4.1.1. Select two random parents  $x_1, x_2$  from current generation.  
Higher fitness means higher probability of selection.
    - 4.1.2. Cross these parents, child is created.
    - 4.1.3. With probability  $M$ , mutate this child.
    - 4.1.4. Apply  $F$  to this child.
  - 4.2. Set all children as population of next generation.

Zdrojový kód 1: Pseudokód genetického algoritmu

### 3.1.1 Inicializace

Základem tohoto algoritmu je populace jedinců. Každý jedinec v populaci představuje svým genotypem potenciální řešení problému (ať už dobré nebo špatné). [1, 4] V případě TSP pro 8 měst může být jedinec reprezentován např. jako sekvence měst, která budou postupně navštívena. Jedinec  $x = \{2, 4, 6, 8, 1, 3, 5, 7\}$  tedy nejdříve navštíví města označená sudými čísly a teprv poté města označená lichými čísly.



Obrázek 41: Někteří z vygenerovaných jedinců GA pro TSP <sup>1</sup>

Na počátku běhu algoritmu dojde k náhodnému vygenerování  $K$  jedinců – ti budou tvořit populaci první generace. Počet jedinců  $K$  v populaci je volen na základě složitosti

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

řešeného problému. Čím vyšší je  $K$ , tím déle bude zpravidla běh algoritmu trvat, avšak tím stabilnější bude a tím spíše najde lepší řešení. [1, 4]

Přestože každý z jedinců představuje potenciální řešení problému, nejspíše žádné z nich nebude dostatečně dobré, protože všechna byla pouze vygenerována náhodně. Na obrázku 41 jsou 3 z vygenerovaných jedinců, kteří v řešeném TSP urazí 13,1 km, 12,6 km a 12,5 km, což je pro daný počet měst poměrně hodně.

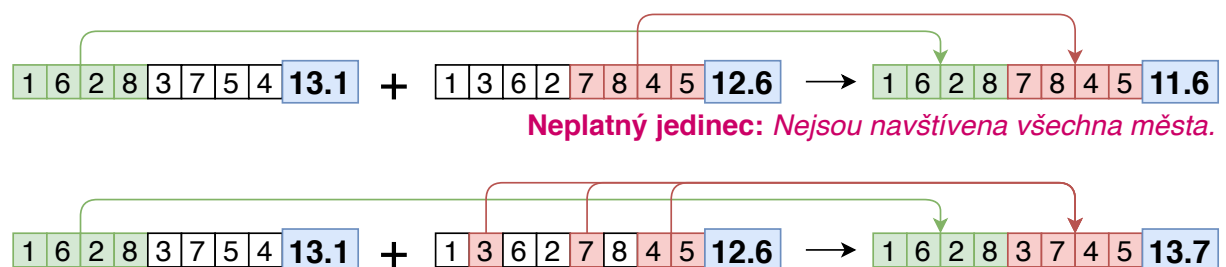
### 3.1.2 Selekcce

Přirozený výběr je založen na skutečnosti, že úspěšnější (silnější) jedinci mají vyšší pravděpodobnost předat své geny a rozmnožit se. Pro napodobení tohoto konceptu v genetickém algoritmu je třeba definovat funkci  $F$ , která dokáže posoudit kvalitu jedince – tzv. fitness. [4]

Podoba této funkce je silně závislá na řešeném problému. V případě TSP jí může být např. převrácená hodnota délky cesty, kterou jedinec urazí. Čím kratší cesta, tím vyšší fitness jedince a tím vyšší pravděpodobnost na předání genů do další generace.

### 3.1.3 Křížení

Poté, co byli vybráni jedinci k předání genů (rodiče) může proběhnout jejich zkřížení. To spočívá v prostém zkombinování genů obou rodičů. Způsob tohoto kombinování opět závisí na typu řešeného problému. Je třeba zajistit, aby výsledkem křížení byl vždy platný jedinec, popř. neplatné jedince dodatečně opravit. [4]



Obrázek 42: Příklad křížení jedinců GA pro TSP <sup>1</sup>

Na obrázku 42 je uveden příklad křížení jedinců při řešení TSP. První způsob křížení vytvoří potomka tak, že první polovinu genů vezme z prvního rodiče a druhou polovinu genů z druhého rodiče. Tímto způsobem však může vzniknout jedinec, který některá

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

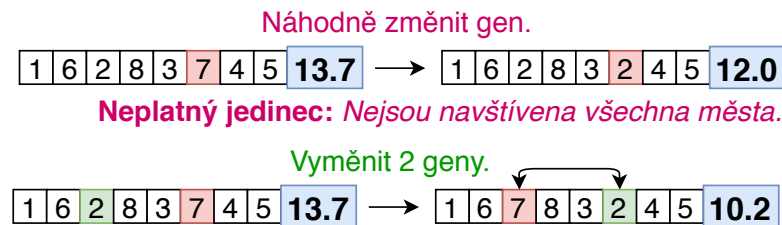
města navštíví vícekrát a jiná ani jednou. Ten však nepředstavuje platné řešení problému, protože TSP vyžaduje, aby všechna města byla navštívena.

Druhý ze způsobů křížení vezme polovinu genů z prvního rodiče a k nim připojí chybějící geny z druhého rodiče. Tímto způsobem tak vznikne vždy platný potomek.

Křížení je opakováno dokud není vytvořeno K potomků tvořících další generaci. [4]

### 3.1.4 Mutace

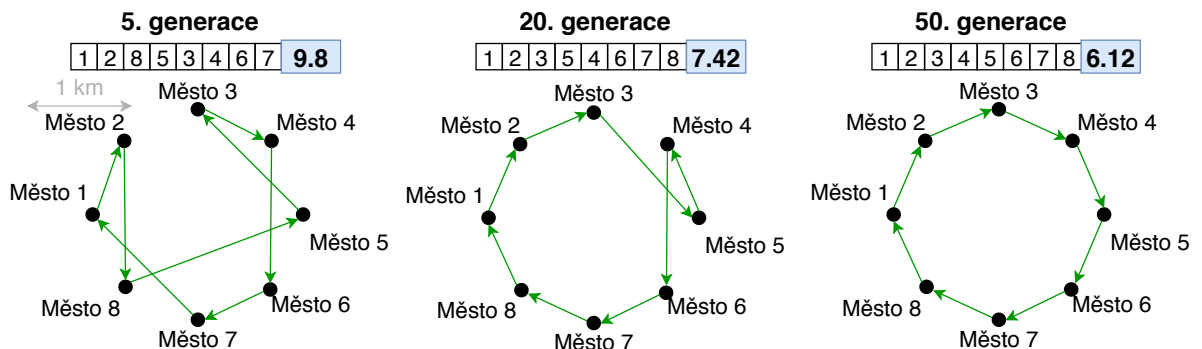
Nyní, když existuje další generace jedinců, je vhodné zajistit vyšší rozmanitost populace. Toho lze docílit nenulovou šancí každého jedince pro mutaci – dojde k náhodné modifikaci jeho genů. [4]



Obrázek 43: Příklad mutace jedince GA pro TSP <sup>1</sup>

Výše pravděpodobnosti opět závisí na řešeném problému. Obvykle se však volí v řádu jednotek procent. Pravděpodobnost může být také proměnlivá, např. v závislosti na podobnosti rodičů – čím podobnější rodiče, tím vyšší pravděpodobnost mutace. [4]

Protože mutace je náhodná změna, dá se očekávat, že ve většině případů povede ke zhoršení kvality jedince. Není však vyloučené, že se kvalita jedince i zvýší. Když se tak stane, takový jedinec bude mít vyšší pravděpodobnost na předání svých genů, čímž dojde i ke zvýšení průměrné kvality celé populace. [4]



Obrázek 44: Hledání řešení TSP pomocí GA <sup>1</sup>

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

## 4 POUŽITÉ TECHNOLOGIE

### 4.1 TypeScript

TypeScript (<https://www.typescriptlang.org/docs>) je open-source nádstavba JavaScriptu poskytující prvky objektového programování – např. třídy a statické typování. Validní JS je zároveň validním TS, a proto lze oba jazyky libovolně kombinovat. [28]

#### 4.1.1 React

React (<https://reactjs.org/docs>) je javascriptová knihovna pro vytváření UI. S využitím syntaxe JSX (resp. TSX) umožňuje zapisovat javascriptový (resp. typescriptový) kód deklarativně v podobě HTML tagů. Veškeré změny stavu aplikace jsou tak automaticky vykresleny. [19]

```
const MyComponent = () => {
  const [count, setCount] = React.useState(0)

  return (
    <button onClick={() => setCount(count + 1)}>{count}</button>
  )
}
```

Zdrojový kód 2: Ukázka práce s knihovnou React

#### 4.1.2 Styled components

Styled components (<https://styled-components.com/docs>) je knihovna umožňující psaní CSS kódu přímo k definici React komponenty. [18]

```
const Button = Styled.button`
  background-color: green;
`
...
<Button>Odeslat</Button>
```

Zdrojový kód 3: Ukázka práce s knihovnou Styled components

## 4.2 Python

Python (<https://www.python.org/doc>) je interpretovaný, dynamicky typovaný skriptovací a programovací jazyk. Podporuje procedurální, objektové a částečně i funkcionální paradigma. Existují různé implementace, např. CPython (v C) nebo JPython (v Javě). Nabízí vysoké pohodlí pro programátora, ovšem na úkor výkonu. Umožňuje však napsat kritické části v nižším jazyku (např. v C) a tento kód pak připojit k programu jako klasický modul v Pythonu. [11]

### 4.2.1 Flask

Flask (<https://flask.palletsprojects.com>) je webový open-source microframework napsaný v jazyce Python. Mikroframework poukazuje na minimalistické jádro, které lze rozšiřovat velkým počtem rozšíření (extensions). [29]

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "Hello World!"
```

Zdrojový kód 4: Ukázka práce s microframeworkem Flask

### 4.2.2 Astropy

Astropy (<https://docs.astropy.org>) je kolekce open-source balíčků napsaných v jazycích C a Python. Primárním účelem této kolekce (knihovny) je poskytovat klíčové funkce a nástroje potřebné pro vytváření astronomického nebo astrofyzikálního software. [16]

```
from astropy import constants as const
from astropy import units as u

print(const.c.to(u.km / u.s)) # 299792.458 km / s
```

Zdrojový kód 5: Ukázka práce s knihovnou Astropy

### 4.2.3 LightKurve

LightKurve (<https://docs.lightkurve.org>) je dalším z open-source balíčků pro programy v jazyce Python. Jedná se o nástavbu nad Astropy specializovanou pro práci se světelnými křivkami a TPF z misí Kepler a TESS. [17]

```
import lightkurve as lk
from astropy import units as u

tpf = lk.search_targetpixelfile("Kepler-10", quarter=4).download()
lc = tpf.to_lightcurve(aperture_mask=tpf.pipeline_mask).flatten()
lc.fold(period=0.837 * u.day).plot()
```

Zdrojový kód 6: Ukázka práce s knihovnou LightKurve

### 4.2.4 Keras

Keras (<https://keras.io/api>) je vysokoúrovňová nástavba nad TensorFlow pro práci s umělými neuronovými sítěmi. Poskytuje jednoduché API a možnost rozšiřitelnosti. [21]

```
model = Sequential([
    Conv2D(16, kernel_size=(3, 3), input_shape=(64, 64)),
    MaxPooling2D(),
    Flatten(),
    Dense(64, activation=tf.nn.relu),
    Dropout(0.3),
    Dense(2, activation=tf.nn.softmax)
])

model.compile(optimizer=tf.keras.optimizers.RMSprop(), loss=MSE())
```

Zdrojový kód 7: Tvorba neuronové sítě v Keras

### 4.2.5 MongoEngine

MongoEngine (<http://docs.mongoengine.org>) je objektově-dokumentové mapování (ODM) mezi jazykem Python a databází MongoDB. Poskytuje jednoduché deklarativní API a umožňuje definovat databázová schémata včetně jejich validace. [24]

```

class Article(Document):
    title = StringField(required=True)
    content = StringField(required=True, min_length=50)
    tags = ListField(EmbeddedDocumentField(Tag))
    author = ReferenceField(User)

```

Zdrojový kód 8: Definice schématu v MongoEngine

### 4.3 MongoDB

MongoDB (<https://docs.mongodb.com>) je multiplatformní dokumentová NoSQL databáze. Ve verzi zdarma přes cloudovou službu MongoDB Atlas poskytuje kapacitu 512 MB. Kromě jednoduchých CRUD operací nabízí i robustní Aggregation Framework pro složitější dotazy a stejně tak podporuje přístup k datům přes paradigma map-reduce. [23, 9]

```

db.planets.aggregate([
    { $match: { name: { $regex: "^Kepler-[0-9]+[a-z]$" } } },
    { $group: { _id: "$_id",
                meanMass: { $avg: { $multiply: ["$density", "$volume"] } } }
    }
])

```

Zdrojový kód 9: Ukázka Aggregation Frameworku z MongoDB

### 4.4 Socket.io

Socket.IO je protokol umožňující real-time obousměrnou komunikaci mezi klienty a serverem. Implementace existují v různých jazycích včetně JavaScriptu (<https://socket.io/docs/v3>) a Pythonu (<https://python-socketio.readthedocs.io>). [20]

```

# Client:
@socketio.event
def process(task):
    print(f"Data processing from address {task['url']}...")

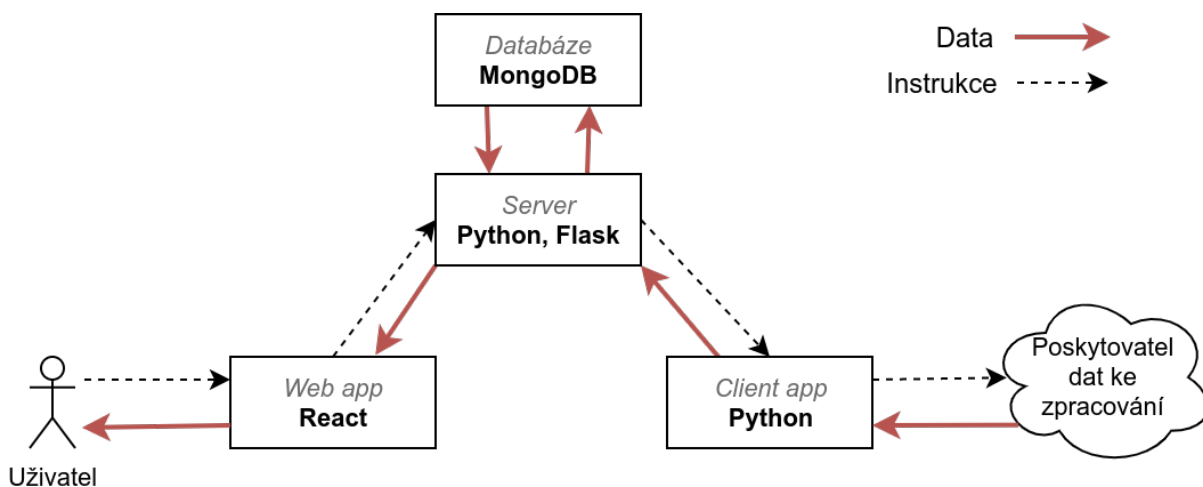
# Server:
task = {type: "LIGHT_CURVE", "url": "http://exoplanetarchive.ipac"}
socketio.emit("process", task)

```

Zdrojový kód 10: Ukázka komunikace pomocí socket.io.



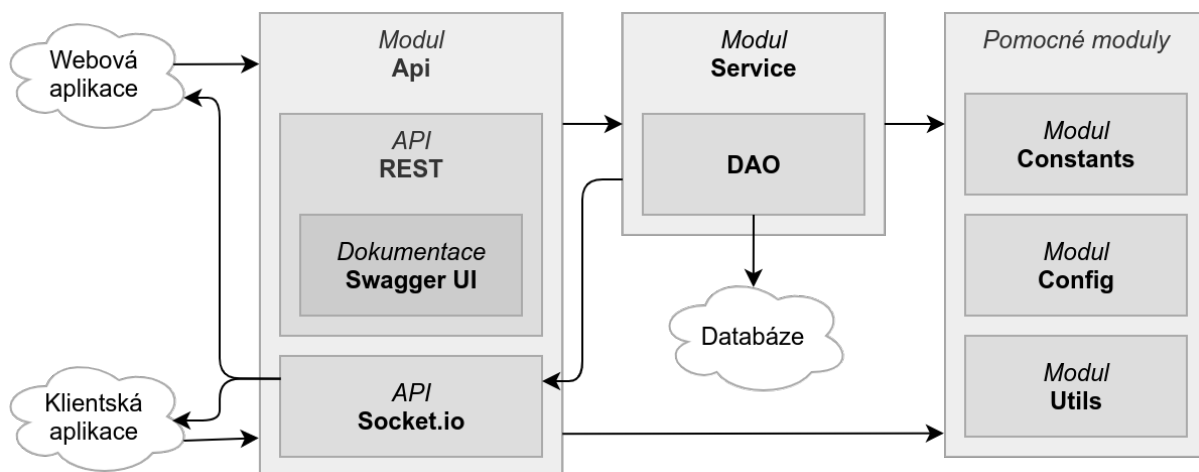
## 5 NÁVRH A VÝVOJ APLIKACE



Obrázek 45: Komponenty projektu a jejich komunikace <sup>1</sup>

### 5.1 Server

Úkolem serveru je přidělování výpočetních úloh připojeným klientským aplikacím, ukládání a načítání persistentních dat ve spolupráci s databází a komunikace s uživatelem skrze webovou aplikaci. Je naprogramován ve webovém frameworku Flask v jazyce Python a struktura jeho zdrojových kódů se dělí na následující moduly:



Obrázek 46: Architektura serveru <sup>1</sup>

- **api** – Definice REST a Socket API a zpracování příchozích požadavků. To zahrnuje autentizaci a autorizaci odesílatele a následné provedení operace v servisní vrstvě.

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

- **config** – Konfigurace serveru (např. připojení k databázi). Základním konfiguračním souborem je `base.cfg`, který je použit automaticky. Bez nutnosti tento soubor měnit je možné vytvořit nový soubor (např. `test.cfg`) s novou konfigurací a pomocí argumentu při spuštění serveru nastavit, aby se použila tato konfigurace (`export ENV=test && ./src/main.py`).
- **constants** – Veškeré konstantní hodnoty, ať už technického či fyzikálního typu.
- **service** – Servisní vrstva obsahující logiku serveru a manipulaci s databází.
- **utils** – Pomocné třídy zapouzdřující specifické funkcionality.

### 5.1.1 REST API

K vytvoření rozhraní je nejdříve třeba nadefinovat příslušné struktury, které rozhraní bude očekávat nebo naopak vracet. Modul `fields` umožňuje určit složky jednotlivých struktur a taktéž pro ně nastavit validační pravidla jako např. maximální délka, datový typ nebo seznam povolených hodnot.

```
credentials = api.model("Credentials", {
    "email": fields.String(required=True, max_length=50),
    "password": fields.String(required=True, max_length=100)
})

user = api.model("User", {
    "name": fields.String(required=True, max_length=30),
    "role": fields.Integer(required=True, enum=UserRole.values())
})
```

Zdrojový kód 11: Vytvoření modelů v REST API.

Následně může dojít k nadefinování koncových bodů rozhraní. Koncovým bodem jsou metody jakékoliv třídy, jež je potomkem `Resource`, přičemž názvy metod třídy odpovídají názvům metod HTTP.

Veškeré vstupní parametry od uživatele jsou automaticky parsovány, při nesprávnosti vstupních parametrů je odeslána chybová odpověď a díky anotacím je také automaticky generována dokumentace REST API v nástroji Swagger UI dostupná na adrese `exoplanets.now.sh/api-docs`. Ukázka dokumentace REST API je umístěna v příloze C na konci dokumentu.

```

@api.route("/login")
class Login(Resource):
    @api.response(HttpStatus.OK, user)
    @api.response(HttpStatus.BAD_REQUEST, "Invalid credentials.")
    @api.expect(credentials)
    def post(self, credentials):
        return user_service.authenticate(credentials)

```

Zdrojový kód 12: Vytvoření koncového bodu v REST API.

Pro redukcí opakujícího se kódu byla v rámci aplikace vytvořena vlastní knihovna. Ta jednotlivé přístupové body vytvoří automatiky na základě nastavené konfigurace. Vyřešena je i automatická autentizace a autorizace dle uživatelských rolí.

```

user_resource = {
    "get": {"auth": EndpointAuth.ANY},
    "get_all": {"auth": EndpointAuth.ANY},
    "post": {"auth": EndpointAuth.ANY},
    "put": {"auth": EndpointAuth.MYSELF},
    "delete": {"auth": UserRole.ADMIN}
}

api.init(service=user_service, model=user, resource_type=user_resource)

```

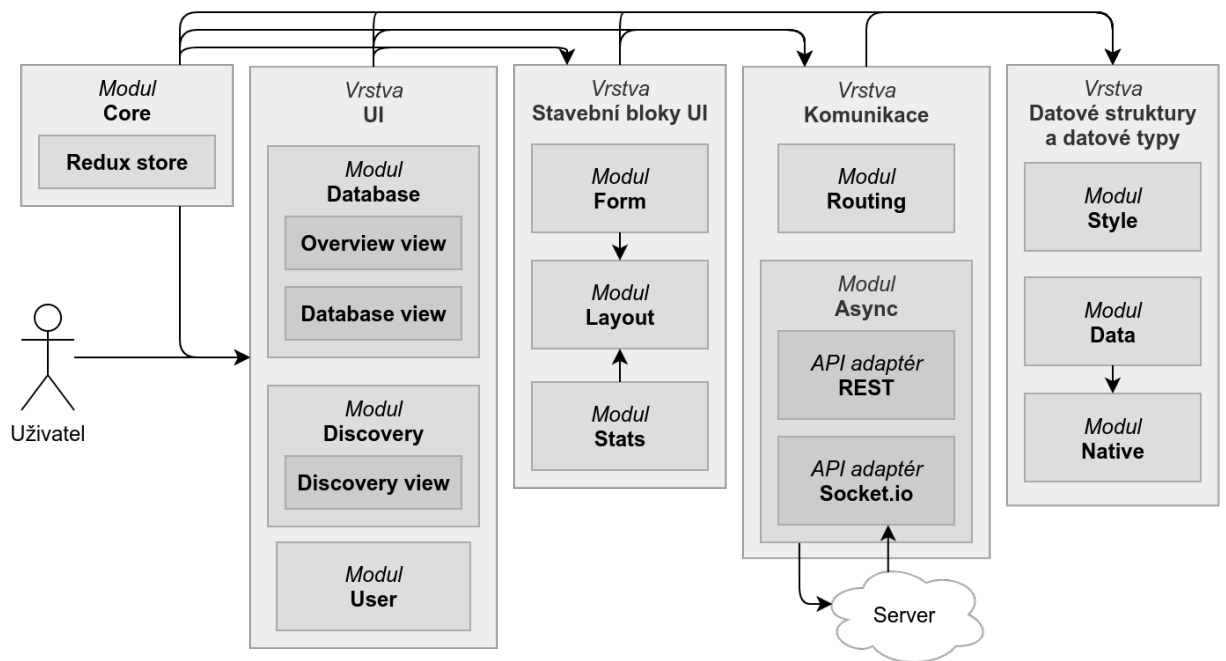
Zdrojový kód 13: Konfigurace zdroje v REST API.

### 5.1.2 Socket.IO API

Protože je třeba v reálném čase synchronizovat webovou a klientskou aplikaci a umožnit serveru iniciovat s těmito aplikacemi komunikaci, server kromě rozhraní *HTTP* poskytuje také rozhraní *Socket.IO*.

## 5.2 Webová aplikace

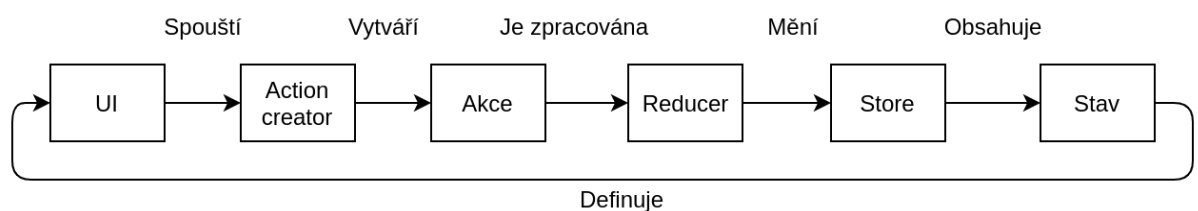
Webová aplikace zprostředkovává interakci s uživatelem. Je napsána v jazyce *TypeScript*, pro tvorbu UI je využita knihovna *React* a pro stylování elementů na stránce knihovna *Styled components*. Zdrojový kód je logicky rozdělen do následujících vrstev a modulů:



Obrázek 47: Architektura webové aplikace <sup>1</sup>

Moduly z vyšších vrstev mohou komunikovat pouze s moduly ze stejné nebo z nižší vrstvy. Tím je zabráněno kruhovým závislostem v kódu.

Ke správě dat je využita architektura a knihovna *Redux* ukládající celý stav aplikace v centralizovaném uložišti (store). Měnit stav tohoto uložišť může pouze funkce *dispatch* jejímž argumentem je akce. Akce je prostý objekt obsahující povinnou položku *type* a libovolné další položky. Pro pohodlnější práci se akce často umísťuje do parametrizovatelné funkce zvané *action creator*, která tuto akci vrací.



Obrázek 48: Princip architektury Redux <sup>1</sup>

Funkce *dispatch* následně zjistí aktuální stav aplikace a společně s akcí v argumentu zavolá funkci *reducer*. Ten na základě aktuálního stavu a akce vrátí stav nový. V poslední řadě dojde k překreslení všech komponent, které jsou napojené na právě změněnou část stavu aplikace – podoba UI je tak vždy aktuální.

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

```

const defaultState = { counter: 0 } // State

const reducer = (state = defaultState, action) => { // Reducer
  switch (action.type) {
    case 'INC': return { ...state, counter: state.counter + action.val }
    case 'RESET': return { ...state, counter: 0 }
  }
}

const store = Redux.createStore(reducer) // Store
const increment = val => ({ type: 'INC', val }) // Action creator
store.dispatch(increment(7)) // Dispatch

```

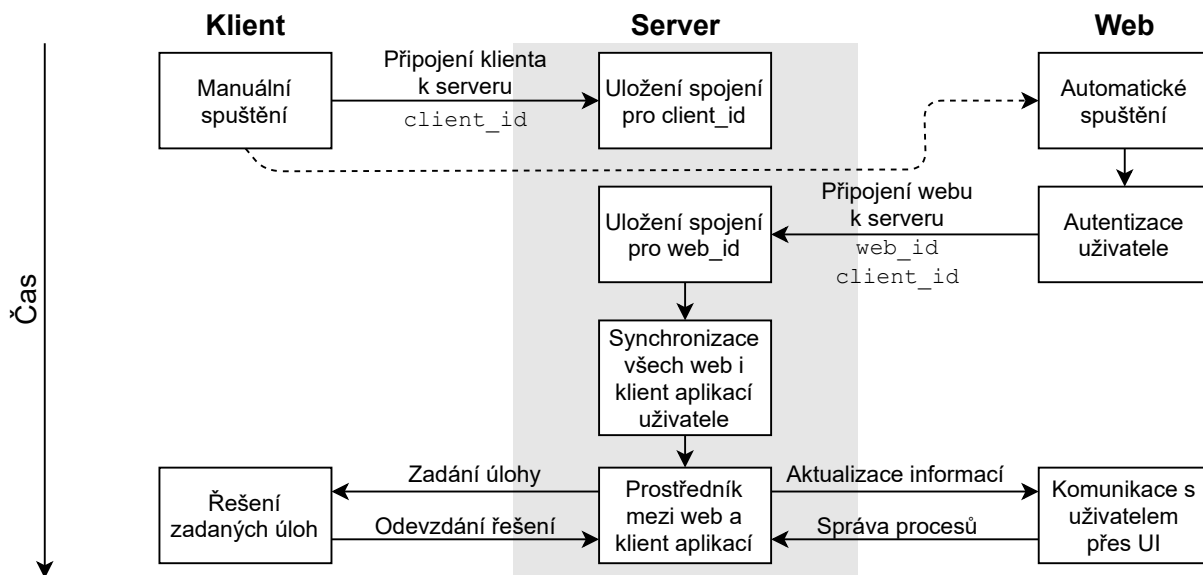
Zdrojový kód 14: Ukázka použití knihovny Redux.

Jednotlivé moduly mají následující funkcionalitu:

- **Core** – Kořenová komponenta `<App />` a router zobrazující dle URL příslušné stránky, dále Redux Store a obstarávání základních funkcionalit aplikace.
- **Database** – Stránky týkající se výpisu/úpravy položek z databáze (např. seznam objevených planet, přidané datasey, ...).
- **Discovery** – Umožňuje uživateli zapojit se do hledání a sledovat své procesy.
- **User** – Stará se o autentifikaci a autorizaci a také o veškerou uživatelskou činnost.
- **Form** – Skládání formulářů z předpřipravených komponent s automatickou validací.
- **Stats** – Usnadňuje vykreslování grafů a komponent se statistickými daty.
- **Layout** – Obsahuje nastýlované a funkční komponenty využívané v celé aplikaci.
- **Routing** – Nabízí pomocné funkce pro práci se směrováním, URL a query řetězci.
- **Async** – Zajišťuje asynchronní operace jako je komunikace se serverem přes *REST* nebo *Socket.io* API. Taktéž animuje `<Loader />` všude tam, kde je to třeba.
- **Style** – Fragmenty stylů využívaných na více místech, media queries pro responzivitu, a konstanty (např. barvy, velikosti, parametry animací, ...).
- **Data** – Zprostředkovává jazykovou lokalizaci, řazení, filtrování a stránkování dat a také práci s fyzikálními veličinami a jednotkami.
- **Native** – Pomocné funkce pro práci s datovými typy (pole, čísla, text, ...).

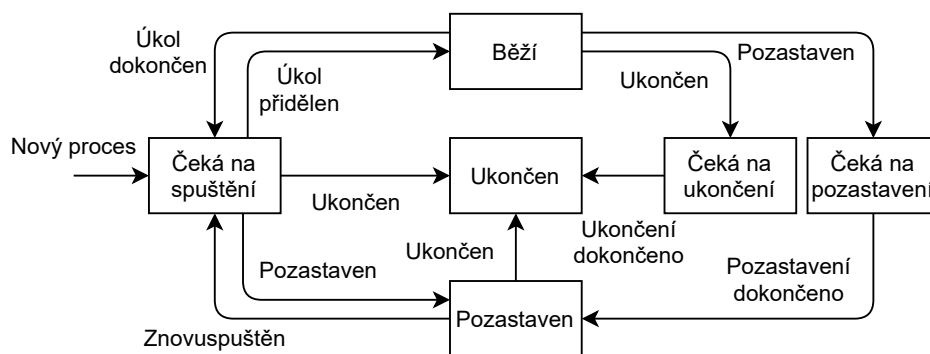
### 5.3 Klientská aplikace

Pro využití většího výpočetního potenciálu počítače není vhodné provádět analýzu dat na straně webové aplikace. Ve webové aplikaci je k dispozici ke stažení klientská aplikace – spustitelný program napsaný v jazyce *Python* optimalizovaný pro zpracování dat.



Obrázek 49: Navázání spojení klientské aplikace <sup>1</sup>

Po spuštění se otevře webová aplikace, ve které uživatel pod svým účtem provede spárování s klientskou aplikací. Následně dojde pomocí *Socket.io* k navázání spojení se serverem, čímž z pohledu uživatele vznikne nový proces ve stavu *Čeká na spuštění*.



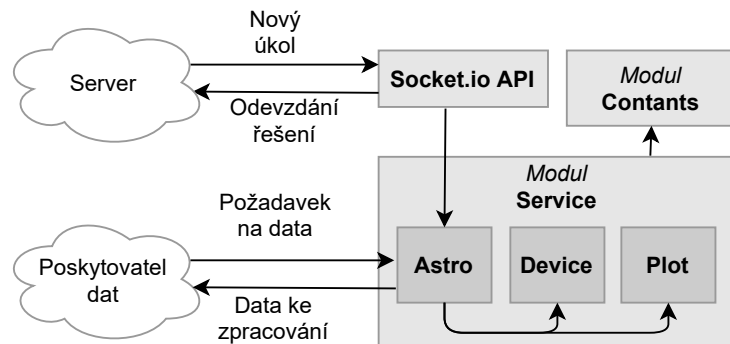
Obrázek 50: Životní cyklus procesu klientské aplikace <sup>1</sup>

V případě, že jsou v databázi některé dosud nezpracované položky z datasetů, server jednu takovou položku přidělí klientské aplikaci a ta ji začne zpracovávat. Po dokončení je řešení odevzdáno serveru, jenž klientské aplikaci přidělí další položku ke zpracování.

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

Tento cyklus se opakuje až do doby, kdy je klientská aplikace pozastavena či ukončena (uživatel tak může učinit z UI webové aplikace).

Do té doby běží program zcela samostatně a uživatel nemusí do průběhu zpracování dat nijak zasahovat. Uživatel může samozřejmě násilně ukončit aplikaci i za běhu. V takovém případě budou dosud zpracovaná data z aktuální položky ztracena.



Obrázek 51: Architektura klientské aplikace <sup>1</sup>

Při komunikaci mezi serverem a klientskou aplikací nedochází k přenosu samotných dat ke zpracování (řádově MiB). Server klientovi posílá pouze metadata k zadávané úloze (např. typ úlohy, adresu URL, ...). Samotná data ke zpracování si už stáhne klient přímo od poskytovatele dat (např. NASA Exoplanet Archive). Server tak není tolik zatížen a stahování většího množství dat je delegováno na jednotlivé klienty.

```

tps = lc_service.download_tps(task["item"]) # Download target pixel.
lc = lc_service.tps_to_lc(tps) # Convert target pixel to light curve.
transits = []

for max_per in LcService.PERIODS:
    pdg = lc_service.get_pdg(lc) # Get periodogram.
    per, t0 = pdg.period_at_max_power, pdg.transit_time_at_max_power
    lc = lc[~pdg.get_transit_mask(per, t0, dur)]
    gv, lv = lc_service.get_views(lc, pdg) # Get global and local view.

    if lc_service.is_planet(gv, lv): # Detect transit using NN.
        transits.append(lc_service.get_transit(pdg, gv, lv))

submit({"solution": {"transits": transits}}) # Submit solution.

```

Zdrojový kód 15: Příklad zpracování světelné křivky klientem

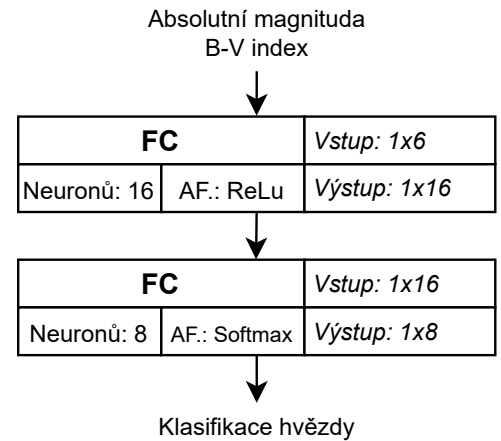
## 5.4 Neuronové sítě

### 5.4.1 Klasifikace hvězdy

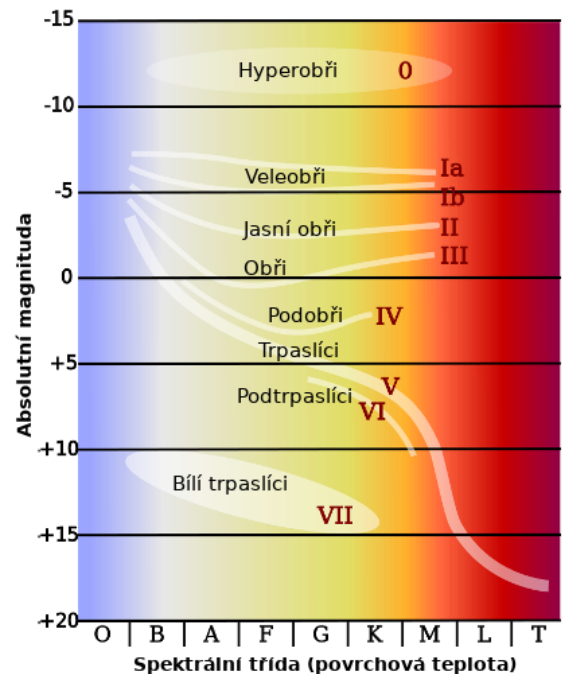
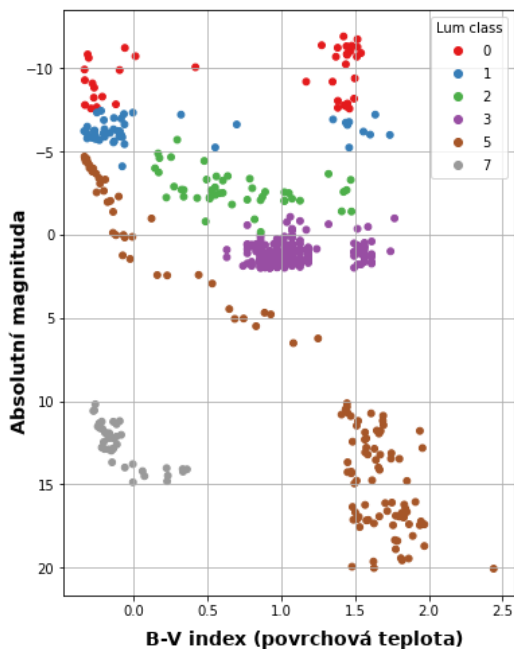
Na základě povrchové teploty a absolutní hvězdné velikosti (magnitudy) lze jednotlivé hvězdy umístit dle Morganovy-Keenanovy klasifikace do jedné z 8 tříd svítivosti. Rozložení těchto tříd je ukázáno v Hertzsprungovu–Russellovu diagramu (HR diagramu).

Zatímco spektrální třída hvězdy je jednoznačně dána dle povrchové teploty hvězdy, třídy svítivosti nejsou lehce separovatelné, a proto je vhodné pro její rozpoznání použít umělou neuronovou síť. Zdrojový

kód pro vytvoření, natrénování a otestování této sítě je umístěn v Jupyter Notebooku `data/lum_class/lum_class.ipynb`. Výstupem sítě je vektor o 8 prvcích, ve kterém největší hodnota značí příslušnost hvězdy do dané kategorie. Např. vektor na výstupu  $(0, 0, 0, 0, 0, 1, 0, 0)$  znamená 6. kategorii, tedy *V* (trpaslík).



Obrázek 52: Architektura neuronové sítě pro klasifikaci hvězd <sup>1</sup>



Obrázek 53: Klasifikace hvězdy – trénovací množina a HR diagram <sup>2</sup>

<sup>1</sup> Vytvořeno autorem v <https://www.draw.io>.

<sup>2</sup> Graf trénovací množiny vytvořen autorem, HR diagram převzat z Wikimedia Commons – public domain.



```

nn = load_model("data/transit/transit.h5") # Load NN from file.
result = nn.predict([[bv, mag]]) # Predict class of star.
lum_class = np.argmax(result[0]) # Get index of most likely class.

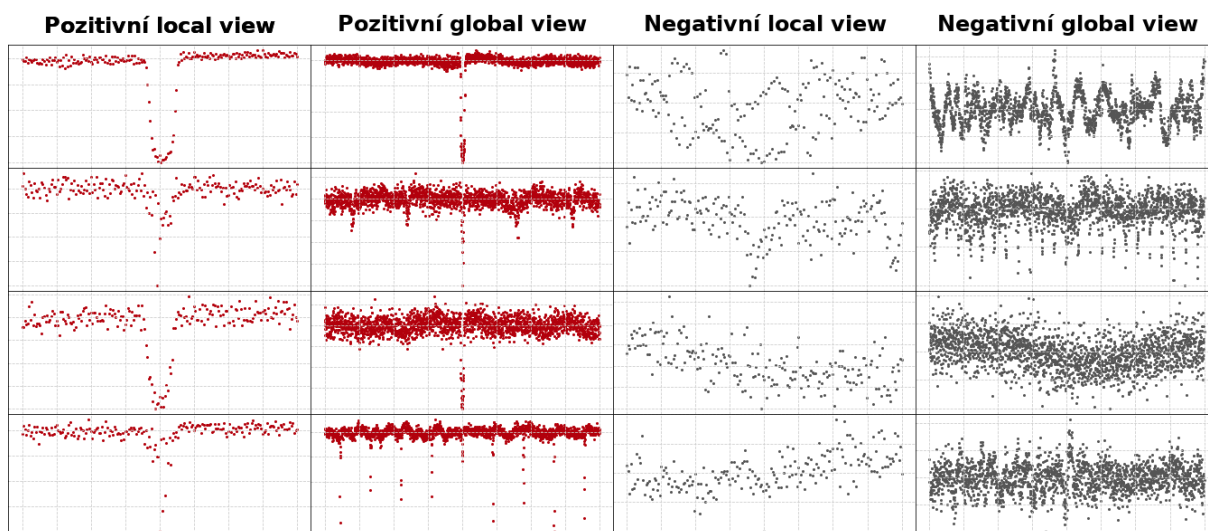
```

Zdrojový kód 16: Použití sítě pro klasifikaci hvězdy

### 5.4.2 Detekce tranzitů exoplanety

V momentě, kdy je nalezena periodická složka ve světelné křivce hvězdy, je třeba ověřit, zda se jedná o tranzit obíhající exoplanety nebo o jiný jev. To je opět problém vhodný pro neuronovou síť.

Síť na vstupu přijme globální a lokální pohled na potenciální tranzit. Výstupem sítě je číslo z intervalu od 0 do 1, přičemž čím větší číslo je, tím s větší jistotou síť považuje vstup za skutečný tranzit planety. Kompletní zdrojový kód sítě se nachází v Jupyter Notebooku `data/transit_nn/transit_nn.ipynb`. Architektura sítě popsána v příloze A.



Obrázek 54: Část trénovací množiny pro detekci tranzitů exoplanet <sup>1</sup>

## 5.5 Databáze

K ukládání persistentních dat je využita databáze *MongoDB*, k níž se na straně serveru přistupuje skrze ODM *MongoEngine*. Než dojde k samotnému připojení, je třeba nadefinovat podobu databáze a validační pravidla jednotlivých kolekcí, tak, jak je to ukázáno v kapitole 4.2.5. Kompletní schéma databáze je umístěno v příloze B.

<sup>1</sup> Vytvořeno autorem, zdroj dat: [25].

```

db = connect(
    app.config["DB_NAME"],
    host=app.config["DB_HOST"],
    username=app.config["DB_USER"],
    password=app.config["DB_PASSWORD"]
)

```

Zdrojový kód 17: Připojení k databázi.

Dokumentová databáze byla zvolena kvůli možnosti snadno vytvářet zanořené dokumenty, což je pro potřeby aplikace důležité. Zároveň však došlo k zanedbání některých dobrých praktik ve prospěch výkonu. Bez těchto ústupů by vyhledávání v databázi obsahující stovky tisíc položek trvalo příliš dlouho, protože by během prohledávání bylo nutno propojovat kolekce mezi sebou.

- U hvězd a planet nejsou datasety uvedeny jako referenční hodnoty (cizí klíče v SQL databázi), ale jako textové řetězce. Vzniká tak duplicita, kterou je nutno udržovat konzistentní na úrovni aplikace.
- Planety nemají vlastní kolekci, ale jsou vnořeny dokumenty u hvězd.

## 5.6 Datasety

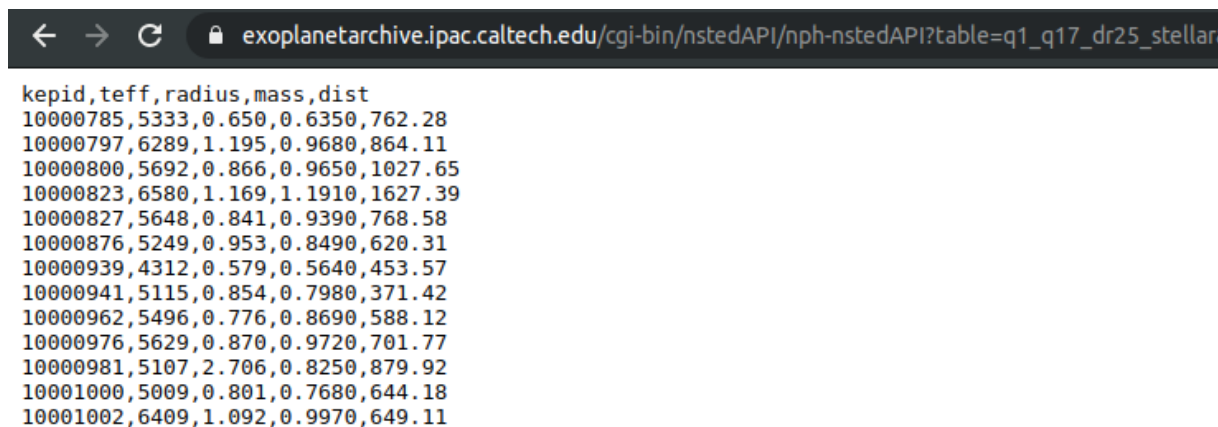
Většina dat se do databáze nekládá manuálně, jelikož je třeba pracovat se stovkami tisíc datových položek. Namísto toho administrátor pouze definuje přístupový bod k nějakému datasetu dostupnému přes webové rozhraní a následně dojde k automatickému zpracování datasetu a všech položek v něm. Některé typy datasetů jsou určeny k jednorázovému uložení do databáze a není nutno je nijak dále zpracovávat. Zpracování jiných je naopak výpočetně náročné a připojené klientské aplikace tak činí postupně položku po položce. Jednotlivé typy datasetů jsou popsány níže.

### 5.6.1 Target pixel files

TPF popsaný v kapitole 1.1.1 je nejčastěji dostupný v souborech s příponou fits. Tyto soubory jsou združovány do datasetů a datasety zase do archivů. Příkladem takového archivu může být *Mikulski Archive for Space Telescopes*, ze kterého aplikace prostřednictvím lighkurve TPF získává.

## 5.6.2 Hvězdy

Informace o vlastnostech hvězd nejsou pro hledání exoplanet nutné, avšak pro spočítání dalších údajů o nalezené exoplanetě je třeba je zahrnout do výpočtů. Datasetsy tohoto typu jsou k dostání přes webová rozhraní nejčastěji ve formátu `csv`.



```
← → ↻ 🔒 exoplanetarchive.ipac.caltech.edu/cgi-bin/nstEDAPI/nph-nstEDAPI?table=q1_q17_dr25_stellar
kepid,teff,radius,mass,dist
10000785,5333,0.650,0.6350,762.28
10000797,6289,1.195,0.9680,864.11
10000800,5692,0.866,0.9650,1027.65
10000823,6580,1.169,1.1910,1627.39
10000827,5648,0.841,0.9390,768.58
10000876,5249,0.953,0.8490,620.31
10000939,4312,0.579,0.5640,453.57
10000941,5115,0.854,0.7980,371.42
10000962,5496,0.776,0.8690,588.12
10000976,5629,0.870,0.9720,701.77
10000981,5107,2.706,0.8250,879.92
10001000,5009,0.801,0.7680,644.18
10001002,6409,1.092,0.9970,649.11
```

Obrázek 55: Část datasetu s vlastnostmi hvězd <sup>1</sup>

Tyto datasetsy neobnáší žádné další složité výpočty, pouze jsou společně s dopočítanými údaji jednorázově uloženy do databáze. Základními údaji, které jsou pro každou hvězdu třeba, jsou (šedou barvou jsou dopočítané údaje):

Název	Zdánlivá magnituda	Rovníkový průměr	Spektrální typ
Vzdálenost od Země	Hmotnost	Absolutní magnituda	Povrchová gravitace
Metalicita	Povrchová teplota	Průměrná hustota	Obyvatelná zóna

Tabulka 10: Údaje o hvězdách ukládané do databáze

Aplikace je postavena flexibilně a umožňuje libovolně namapovat sloupce z datasetů do sloupců v databázi. Datasetsy tak mohou mít pořadí i názvy jednotlivých sloupců libovolné.

## 5.6.3 Planety

Datasetsy s planetami nejsou pro běh aplikace taktéž nutné, protože veškeré informace o exoplanách jsou vypočítány z jiných datasetů. Platí však, že čím více nezávislých zdrojů se na údajích o exoplanetě shodne, tím spíše budou tyto údaje platné. Proto je umožněno ukládat do databáze i datasetsy s údaji o planetách – mohou potvrdit nebo vyvrátit údaje vypočtené v rámci aplikace.

<sup>1</sup> Dataset pochází ze stránek <https://exoplanetarchive.ipac.caltech.edu>.

```

pl_hostname,pl_letter,pl_controvflag,pl_pnum,pl_orbper,pl_orbperlim,pl_orbpern,pl_orbsm
_radj,pl_radjlim,pl_radn,pl_dens,pl_denslim,pl_densn,pl_facility
HD 10180,f,0,6,122.74400000,0,2,0.492900,0,2,0.119000,0,2,,1,0.07220,0,2,Msini,,,0,,0
HD 10180,g,0,6,604.67000000,0,2,1.427000,0,2,0.263000,0,2,,1,0.07320,0,2,Msini,,,0,,0
HD 10180,h,0,6,2205.00000000,0,2,3.381000,0,2,0.095000,0,2,,1,0.20660,0,2,Msini,,,0,,0
HD 10442,b,0,1,1032.30000000,0,4,2.010000,0,4,0.132000,0,4,,0,1.48700,0,4,Msini,,,0,,0
HD 10647,b,0,1,989.20000000,0,2,2.015000,0,2,0.150000,0,2,,0,0.94000,0,2,Msini,,,0,,0
HD 10697,b,0,1,1075.69000000,0,3,2.140000,0,3,0.104300,0,3,,0,6.38300,0,3,Msini,,,0,,0
HD 11506,b,0,2,1622.10000000,0,5,2.900000,0,5,0.374300,0,5,,0,4.83000,0,5,Msini,,,0,,0
HD 11506,c,0,2,223.41000000,0,3,0.774000,0,3,0.193000,0,3,,0,0.40800,0,3,Msini,,,0,,0
HD 11977,b,0,1,711.00000000,0,2,1.930000,0,3,0.400000,0,2,,0,6.54000,0,3,Msini,,,0,,0
HD 11964,b,0,2,1945.00000000,0,3,3.160000,0,2,0.041000,0,3,,0,0.62200,0,3,Msini,,,0,,0
HD 11964,c,0,2,37.91000000,0,1,0.229000,0,1,0.300000,0,1,,0,0.07880,0,1,Msini,,,0,,0
HD 12661,b,0,2,262.70862000,0,5,0.840000,0,5,0.380000,0,5,,0,2.43000,0,5,Msini,,,0,,0

```

Obrázek 56: Část datasetu s vlastnostmi planet <sup>1</sup>

Opět se jedná o datasety nejčastěji ve formátu `csv`, které není nutno nijak složitě zpracovávat, pouze uložit do databáze.

Název	Povrchová teplota	Excentricita dráhy	Rovníkový průměr
Velká poloosa	Typ	Hmotnost	Perioda oběhu
Tranzit přes hvězdu	Průměrná hustota	Rychlost oběhu	Podmínky pro život

Tabulka 11: Údaje o planetách ukládané do databáze

## 5.6.4 Názvy

Často se stává, že jedno a to samé těleso je v různých datasetech pod různým označením.

Katalog	Účel	Označení Kepler-10
KIC ( <b>K</b> epler <b>I</b> nput <b>C</b> atalog)	Hledání exoplanet (Kepler)	KIC 11904151
KOI ( <b>K</b> epler <b>O</b> bject of <b>I</b> nterest)	Výběr z KIC	KOI-72
Kepler	Potvrzené exoplanety z KOI	Kepler-10
2MASS ( <b>2</b> Micron <b>A</b> ll-Sky <b>S</b> urvey)	Infra. průzkum oblohy	2MASS J19024305+5014286
GSC ( <b>G</b> uide <b>S</b> tars <b>C</b> atalog)	Pozorování hvězd (Hubble)	GSC 03549-00354
Gaia DR ( <b>G</b> aia <b>D</b> ata <b>R</b> elease)	Měření polohy hvězd (Gaia)	Gaia DR2 2132155017099178624
USNO-B1.0	Pozorování hvězd a galaxií	USNO-B1.0 1402-00324696
UCAC3	Pozorování hvězd	UCAC3 281-142262

Tabulka 12: Pojmenování soustavy Kepler-10 v různých katalozích <sup>2</sup>

K zamezení toho, aby byly tyto položky vedeny jako dvě různé soustavy je nutno aplikaci poskytnout informace o používaných názvech jednotlivých objektů. Právě k tomu slouží tento typ datasetů.

<sup>1</sup> Zdroj: <https://exoplanetarchive.ipac.caltech.edu>.

<sup>2</sup> Zdroj dat: <http://simbad.u-strasbg.fr/simbad/sim-id?Ident=Kepler-10>.

## 5.7 Instalace a spuštění aplikace

Běžící aplikace je dostupná na adrese <https://exoplanets.now.sh>, dokumentace REST API pak na <https://exoplanets-server.herokuapp.com/api-docs>. Pro spuštění aplikace na lokálním počítači je třeba:

Požadavek	Ke stažení
Python 3.8	<a href="https://www.python.org/downloads">https://www.python.org/downloads</a>
Node.js 10	<a href="https://nodejs.org/en/download">https://nodejs.org/en/download</a>
mongoDB 4.0	<a href="https://www.mongodb.com/try/download/community">https://www.mongodb.com/try/download/community</a>

Tabulka 13: Požadavky pro spuštění aplikace

Po nainstalování závislostí je nutno přejít v kořenovém adresáři do podadresáře `scripts` a v něm spustit soubor `build.sh` (Linux), čímž dojde k sestavení serveru, webové i klientské aplikace. Pro sestavení na jiných operačních systémech lze manuálně použít analogické příkazy pro specifický OS, jako v uvedeném skriptu pro Linux.

```
./build.sh  
./run.sh
```

Zdrojový kód 18: Sestavení a spuštění aplikace (Linux).

Sestavená klientská aplikace je v adresáři `client/build`, webová aplikace v `web/build`. Lze ji umístit na vlastní webový server s přístupovým bodem `index.html` nebo ji spustit v předpřipraveném webovém serveru pomocí skriptu `run.sh`.

Komponenta	Adresa/Cesta
Server	<a href="http://localhost:5000">http://localhost:5000</a>
Web	<a href="http://localhost:3000">http://localhost:3000</a>
Klient	<code>client/build</code>
Databáze	<a href="http://localhost:27017">http://localhost:27017</a>

Tabulka 14: Výchozí adresy komponent po spuštění

### 5.7.1 Konfigurace

Pokud není výchozí konfigurace vyhovující, je možno před sestavením některé parametry serveru změnit v souboru `server/config/base.cfg`.

```
PORT=5000  
DATABASE_NAME="exoplanets"  
DATABASE_HOST="mongodb://localhost:27017"
```

```
DATABASE_USER="user"
DATABASE_PASSWORD=""
```

Zdrojový kód 19: Výchozí konfigurace serveru.

Jestliže je třeba více oddělených konfigurací pro různé režimy, lze vytvořit vlastní konfigurační soubory ve tvaru `server/config/mode_name.cfg`. Pro využití této konfigurace je pak třeba před spuštěním serveru nastavit proměnnou prostředí `ENV` na příslušnou hodnotu.

```
export ENV=mode_name
./build.sh
./run.sh
```

Zdrojový kód 20: Spuštění a sestavení aplikace s vlastní konfigurací (Linux).

V souboru `web/src/Async/Constants/Config.ts` je také možno upravit adresu serveru, se kterým bude komunikovat webová aplikace.

```
export default {
  serverUrl: 'http://localhost:5000',
  apiPrefix: '/api'
}
```

Zdrojový kód 21: Výchozí konfigurace webové aplikace.

Stejným způsobem lze upravit i konfiguraci `client/constants/config.py` pro klientskou aplikaci. Specifikováním příslušných argumentů je možno sestavení aplikovat pouze na některé části projektu a ne na všechny zároveň (server, web i klient).

```
./build.sh // Build all.
./build.sh --client // Build only client.
./run --web --server // Run web and server.
```

Zdrojový kód 22: Argumenty spouštěcích skriptů (Linux).

## 5.7.2 Vývoj

Postup výše nám umožní jednorázově sestavit a spustit aplikaci. Pokud je však třeba aplikaci dále vyvíjet, je vhodné ji spustit ve vývojovém režimu. To zajistí výpis chyb do konzole, restart serveru při změně kódu a oddělenou databázi od produkčních dat.

```
./dev.sh --server --web
```

Zdrojový kód 23: Spuštění v režimu pro vývoj (Linux).

### 5.7.3 Testování

V rámci projektu nebyly vytvořeny kompletní testy zcela pokrývající všechny 3 aplikace. Existují však integrační testy pro některé stěžejní části serverové části napsané s využitím knihovny `pytest` a stejně tak pro webovou část s využitím knihovny `jest`.

```
./test.sh --server
```

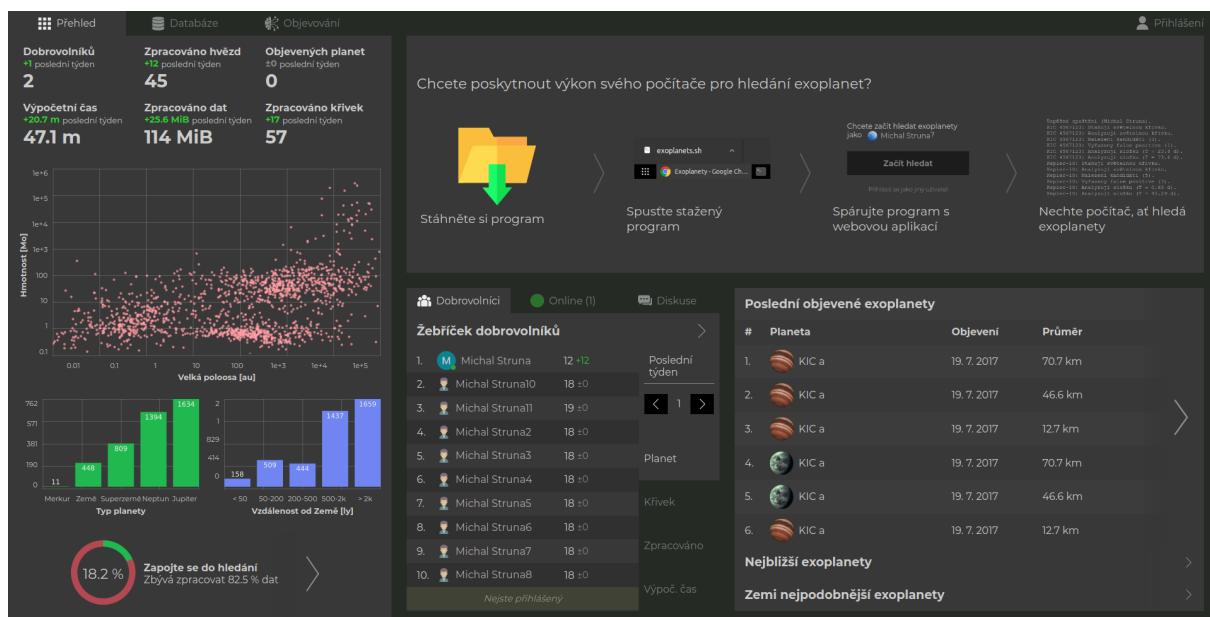
```
./test.sh --web
```

Zdrojový kód 24: Spuštění testů aplikace (Linux).

# 6 ROZVRŽENÍ APLIKACE

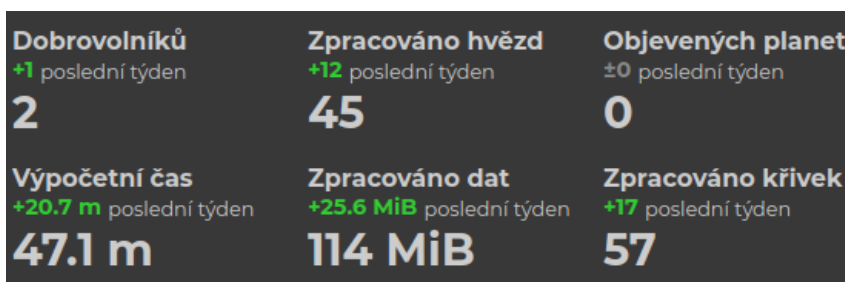
## 6.1 Přehled

Domovská stránka nabízí přehled globálních statistik jako např. počet objevených exoplanet, registrovaných dobrovolníků nebo množství zpracovaných dat. Dále se zde nachází žebříčky exoplanet dle různých kritérií (nejbližší exoplanety, exoplanety nejpodobnější Zemi, ...) a stejně tak i žebříčky dobrovolníků (dle výpočetního času, počtu objevených planet, ...).



Obrázek 57: Stránka Přehled

V rámci aplikace jsou sbírány všemožné statistiky v rámci jednotlivých položek (např. dle data-setu či uživatele). Mimo to zde ale na přehledové



Obrázek 58: Statistika

stránce lze vidět i statistiky globální. U všech ze zobrazených statistik lze vidět jednak aktuální hodnotu a pak také změnu za poslední týden. Statistiky jsou vždy aktuální (nechácejí se) a přesto je zajištěna efektivní práce s nimi.



## 6.2 Databáze

Všechna veřejně dostupná data (hvězdy, exoplanety, datasety, ...) lze procházet na této stránce. Samozřejmostí je stránkování, řazení dle libovolného údaje i filtrování dle libovolného údaje nebo relace (rovná se, je menší, obsahuje, ...). Administrátoři mají navíc možnost některé položky přidávat, upravovat či mazat.

#	Typ	Název	Průměr	Hmotnost	Hustota	Teplota	Vzdálenost	Zář. výkon	Světelná křivka	Planet	Gravit
1	KIC 10020423	Zlutý trpaslík G3V	173 ☉	0.846 ☉	229 $\rho_{\text{Jup}}$	5 636 K	920 ly	3.01 ☉		3	771 $m_{\text{Jup}}$
	KIC a	Jupiter	70.7 ☉	19 774 ☉	307 $\rho_{\text{Jup}}$	940 °C	0.071 au	7.45 d		Vyloučen	38.7 $m_{\text{Jup}}$
	KIC a	Jupiter	46.6 ☉	7 186 ☉	391 $\rho_{\text{Jup}}$	940 °C	0.071 au	7.45 d		Vyloučen	32.5 $m_{\text{Jup}}$
	KIC a	Jupiter	12.7 ☉	311 ☉	832 $\rho_{\text{Jup}}$	940 °C	0.071 au	7.45 d		Vyloučen	18.9 $m_{\text{Jup}}$
2	KIC 10418224	Zlutobílý trpaslík F6V	4.36 ☉	1.52 ☉	25.8 $\rho_{\text{Jup}}$	6 490 K	2 186 ly	19 ☉		1	21.9 $m_{\text{Jup}}$
	KIC a	Jupiter	27.7 ☉	2 051 ☉	529 $\rho_{\text{Jup}}$	1 520 °C	0.081 au	6.87 d		Vyloučen	26.1 $m_{\text{Jup}}$
3	KIC 5794240	Oranžový trpaslík K8V	114 ☉	0.587 ☉	565 $\rho_{\text{Jup}}$	3 793 K	337 ly	1.29 ☉		2	125 $m_{\text{Jup}}$

Obrázek 59: Stránka Databáze

Přidávání hvězd a planet do databáze se neprovádí manuálně, nýbrž prostřednictvím datasetů. Jednotlivé typy datasetů jsou popsány v kapitole 5.6. Každý typ datasetu má svou vlastní podobu formuláře, ve kterém je možné definovat přístup k jednotlivým položkám datasetu.

### Nový dataset

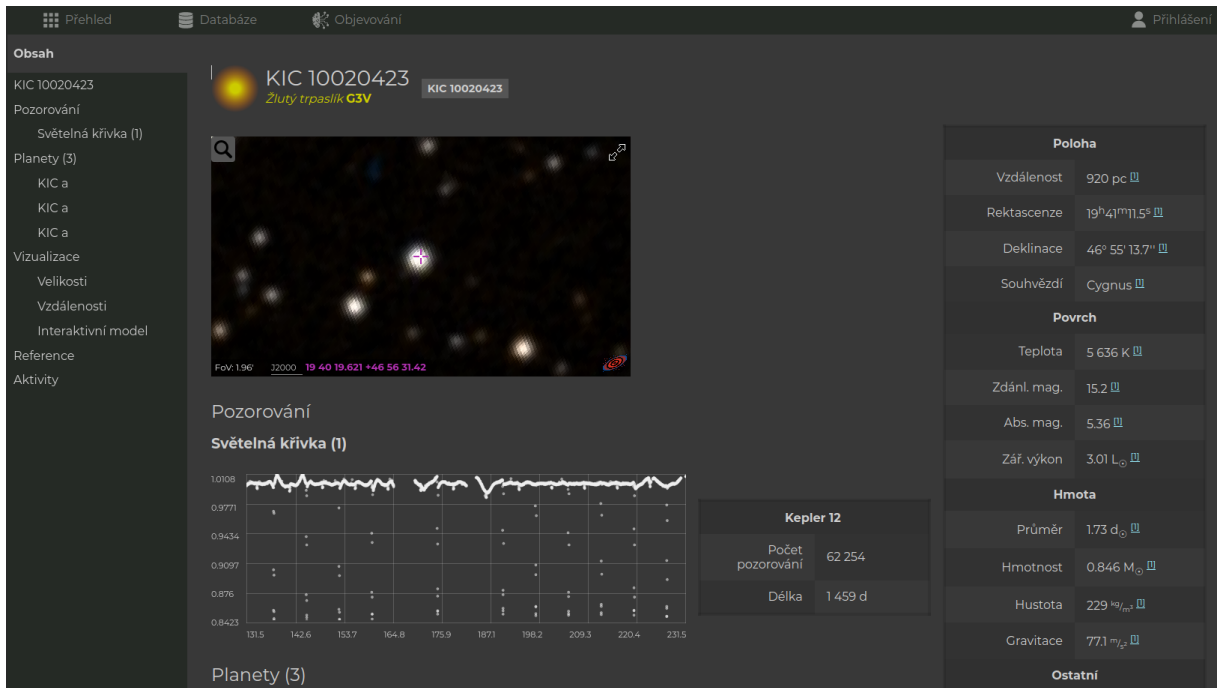
Přehled	Název	Typ	Priorita
	Kepler Stellar	Hvězdy	Vysoká
Pole	Items getter	Item getter	
	https://exoplanetarc		
	Název	Teplota	Průměr
	KIC {kepid}	{teff}	{radius*2}
Hmotnost	Rektascenze	Deklinace	
{mass}	{ra}	{dec}	
Vzdálenost	Zdánl. mag.	Metalicita	
{dist}	{kepmag}	{feh}	

➔

Obrázek 60: Formulář pro nový dataset

## 6.3 Detail systému

Tato stránka shrnuje všechny známé informace o systému. Kromě hodnot veličin hvězdy a případných planet jsou zde zaznamenána všechna pozorování, vizuální porovnání velikostí a vzdáleností, seznam referencí a také interaktivní mapa oblohy.



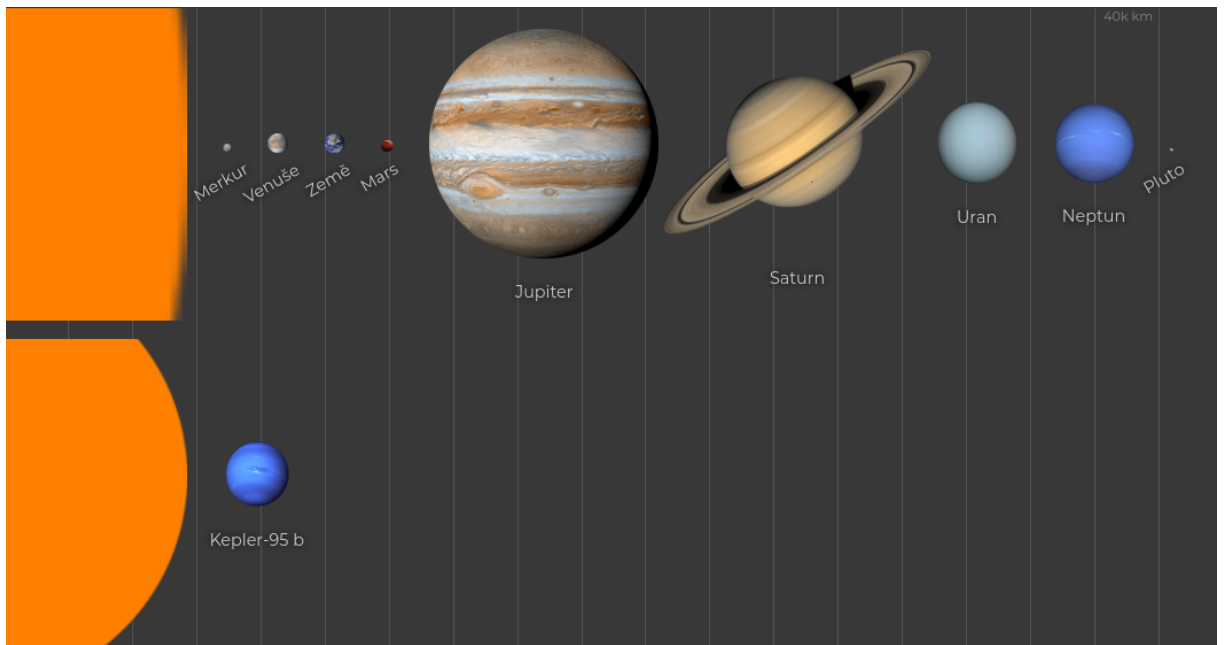
Obrázek 61: Stránka Detail systému

U každé z objevených planet v systému je kromě tabulky zobrazen i lokální a globální pohled na tranzit.



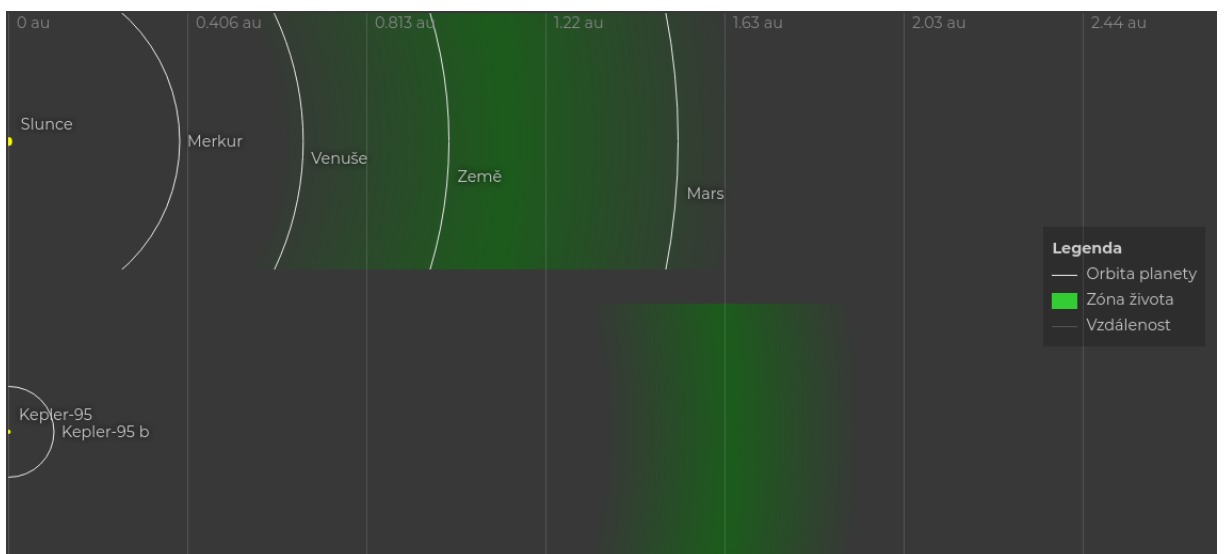
Obrázek 62: Detail planety

Komponenta pro vizuální porovnání velikostí planet a hvězd zobrazuje jednak vybraný planetární systém a jednak naši sluneční soustavu. Velikosti hvězd i planet jsou v realistickém poměru. Vzdálenosti jsou však kvůli enormním velikostem z tohoto diagramu vypuštěny.



Obrázek 63: Provnání velikostí

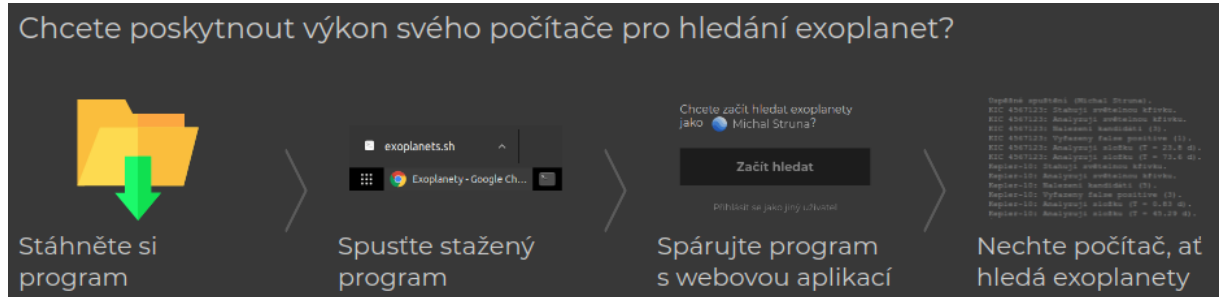
Komponenta pro vizuální porovnání vzdálenosti planet na druhou stranu zanedbává velikosti a v realistickém měřítku zobrazuje pouze vzdálenosti. Pro porovnání je zde opět sluneční soustava. Kromě toho se zde nachází i vyznačení tzv. zóny života. To je vzdálenost od hvězdy, ve které je vhodná rovnovážná teplota pro existenci vody v kapalném stavu.



Obrázek 64: Provnání vzdáleností

## 6.4 Objevování exoplanet

V horní části je umístěn odkaz pro stažení klientské aplikace jakožto i stručný návod pro její spuštění. Níže je zobrazen seznam spuštěných klientských aplikací (procesů) uživatele.



Obrázek 65: Návod pro spuštění klientské aplikace

U každého procesu jsou základní informace jako např. operační systém, doba běhu nebo kompletní log událostí z procesu zpracování dat.

<b>msnb</b>	Čeká na data 12 m 33 s	20.12:41, 716	Spojení navázáno (Michal Struna).
OS Ubuntu 20.04	CPU Intel Core i5-8300H CP...	20.12:41, 854	Stahuji target pixel KIC 11904151.
Analyzovaných křivek 16	Objevených planet 2	20.12:44, 043	Analyzuji světelnou křivku KIC 11904151.
Pozastavit	Ukončit	20.12:56, 442	<b>Nalezena perioda 0.837 d (planeta).</b>
		20.13:03, 166	Nalezena perioda 16.35 d (false positive).
		20.13:10, 149	<b>Nalezena perioda 45.29 d (planeta).</b>
		20.13:14, 598	Nalezena perioda 128.6 d (false positive).
		20.13:14, 628	Stahuji target pixel KIC 10000800.
		20.13:18, 526	Analyzuji světelnou křivku KIC 10000800.
		20.13:25, 833	Nalezena perioda 1.569 d (false positive).

Obrázek 66: Informace o procesu

## 6.5 Autentizace

Komponenty řešící autentizaci uživatele nevyžadují samostatnou stránku, ale jsou kdykoliv dostupné v pravém horním rohu aplikace, kde je lze kliknutím na tlačítko rozbalit. Vždy jsou viditelné pouze ty komponenty, které daná uživatelská role vyžaduje. Jejich kompletní výčet je uveden v tabulce 15.

Kromě toho může každý uživatel do svého profilu vyplnit i některé osobní údaje, na jejichž základě lze vybudovat statistiku porovnávající působení uživatelů v aplikaci na základě jejich věku, pohlaví či země.

Facebook Google

Nebo

Email

Heslo

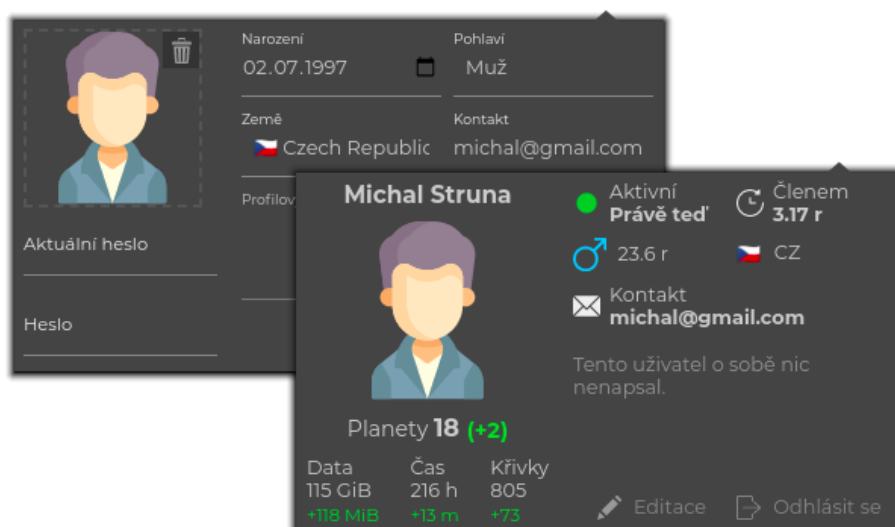
Přihlásit se

Zaregistrovat se Resetovat heslo

Obrázek 67: Přihlašovací formulář

Komponenta	Role	Popis
Přihlašovací formulář	Nepřihlášený	Umožňuje uživateli přihlásit se do aplikace. Nabízí možnost autentizace přes email a heslo nebo i přes Facebook či Google.
Registrační formulář	Nepřihlášený	Slouží pro registraci nových uživatelů. Opět umožňuje reistraci přes email a heslo nebo přes externí služby.
Formulář pro zapomenuté heslo	Nepřihlášený	Uživatelé si mohou na svůj email poslat odkaz pro nastavení nového hesla.
Uživatelský panel	Přihlášený	Informace o aktuálně přihlášeném uživateli včetně tlačítka pro odhlášení.

Tabulka 15: Autentizační komponenty



Obrázek 68: Profil a editace uživatele

## ZÁVĚR

V rámci této diplomové práce byla vytvořena trojice aplikací – webová, klientská a serverová – s jejichž využitím jsou uživatelé schopni poskytovat výpočetní výkon svých počítačů pro detekci a analýzu exoplanet. Úspěšnost při řešení jednotlivých typů úloh je popsána níže. Kompletní výsledky dosažené při verifikaci aplikace pak v příloze D.

Kategorie	Pokusů	Přesnost
Klasifikace hvězd	130	97,69 %
Pozitivní tranzity	22	88 %
Negativní tranzity	60	100 %
Výpočet period planet	22	100 %
Výpočet průměrů planet	22	93,06 %
Výpočet hmotností planet	18	39,57 %

Tabulka 16: Verifikace výsledků aplikace

Aplikace je velice odolná vůči falešně pozitivním tranzitům (žádný takový nebyl zaznamenán). Avšak objevilo se několik falešně negativních tranzitů, zejména pokud jde o víceplanetární systémy, tam je nejčastěji detekován pouze nejsilnější tranzit. Výpočet period oběhů a průměrů planet je taktéž na velice slušné úrovni. Hmotnost planet ovšem nelze z tranzitní metody vypočítat. V rámci aplikace docházelo pouze k jednoduchému odhadu hmotnosti na základě velikosti planety.

Pro zahájení hledání exoplanet je po uživateli vyžadována registrace. Ta je možná buď přes email nebo přes některé ze sociálních sítí. Poté je uživateli umožněno stáhnutí klientské aplikace a zahájení hledání. Další aktivita ze strany uživatele již není třeba.

V případě zájmu je ovšem uživateli umožněno z webové aplikace procházet všechna zveřejněná data, jež jsou výsledkem onoho distribuovaného hledání exoplanet. Webová aplikace je doplněna o grafy, tabulky a statistiky o planetách, datasetech či uživateli. Není však řešena podpora pro tablety a mobily (resp. responzivita).

Klientská aplikace je malý spustitelný soubor dostupný ke stažení z prostředí webové aplikace. Po jeho spuštění dojde ke spárování s uživatelem a k zahájení výpočtů. Program je k dostání jako `exe` nebo `sh` (je tedy dostupný pro Windows a pro Linux).

Aplikace v současné době detekuje exoplanety pouze na základě tranzitní metody. Pro zvýšení četnosti objevů a zpřesnění vypočítaných veličin by tento proces bylo vhodné

doplnit i dalšími metodami. U většiny ostatních metod však nebyly autorem nalezeny dostatečně velké a volně přístupné datasety.

Dalším zlepšením do budoucna by mohlo být zvýšení granularity úloh. Analýza položky z datasetu se světelnými křivkami trvá na průměrném osobním počítači řádově jednotky minut. V případě, kdy uživatel násilně ukončí aplikaci uprostřed výpočtu by bylo vhodné nepřijít o dosud vypočítaná data a v průběhu samotného výpočtu je ukládat i na server, aby ve výpočtech mohl případně pokračovat jiný uživatel. Na druhé straně, příliš velká granularita by naopak způsobila zbytečně velké množství přenesených dat přes síť.

V neposlední řadě by také mohlo dojít ke zlepšení odhadu hmotností exoplanet. K tomu by bylo možno použít techniku popsanou v kapitole 1.1.5 využívající umělou neuronovou síť. Odhad hmotnost planety na základě více veličin by mohl být spolehlivější, než-li odhad pouze na základě velikosti.

## POUŽITÁ LITERATURA

- [1] CARPINTERO D. D. MELITA M. D. An alternative stable solution for the Kepler-419 system, obtained with the use of a genetic algorithm. *Astronomy & Astrophysics* 620(A88). [online]. 2018. [cit. 22. 11. 2020]. Dostupné z: <https://arxiv.org/abs/1810.06769>
- [2] DATTIO, Anne. Identifying Exoplanets with Deep Learning. II. Two New Super-Earths Uncovered by a Neural Network in K2 Data. *The Astronomical Journal* 157(5). [online]. 9. 4. 2019. [cit. 23. 10. 2020]. Dostupné z: <https://iopscience.iop.org/article/10.3847/1538-3881/ab0e12>
- [3] DOLEŽEL, Petr. Úvod do umělých neuronových sítí. *Univerzita Pardubice, Fakulta elektrotechniky a informatiky*. 2016. [cit. 9. 10. 2020]. ISBN 978-80-7560-022-6
- [4] CHIUNG, Moon, JONGSOO, Kim, GYUNGHYUN, Choi, YOONHO, Seo. An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research*. [online]. 2002. [cit. 28. 6. 2021]. Dostupné z: [https://doi.org/10.1016/S0377-2217\(01\)00227-2](https://doi.org/10.1016/S0377-2217(01)00227-2)
- [5] KHAN, Salman. A Guide to Convolutional Neural Networks for Computer Vision. *Morgan & Claypool*. 2018. [cit. 26. 10. 2020]. ISBN 781681730226
- [6] LOVIS, Christophe, FISCHER, Debra A. Radial Velocity Techniques for Exoplanets. *University of Arizona Press*. [online]. 2011. [cit. 25. 12. 2019]. Dostupné z: [https://www.researchgate.net/publication/253789798\\_Radial\\_Velocity\\_Techniques\\_for\\_Exoplanets](https://www.researchgate.net/publication/253789798_Radial_Velocity_Techniques_for_Exoplanets)
- [7] MARCY, Geoffrey. The planet around 51 Pegasi. *The astrophysical journal* 481(2). [online]. 1997. [cit. 29. 12. 2019]. Dostupné z: <https://iopscience.iop.org/article/10.1086/304088>
- [8] MOUTOU, Claire, PONT, Frédéric. Detection and characterization of extrasolar planets: the transit method. *Strasbourg: Observatoire astronomique de Strasbourg et Société Française d'Astronomie et d'Astrophysique*. [online]. 2006. [cit. 8. 10. 2020]. Dostupné z: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.4155&rep=rep1&type=pdf>



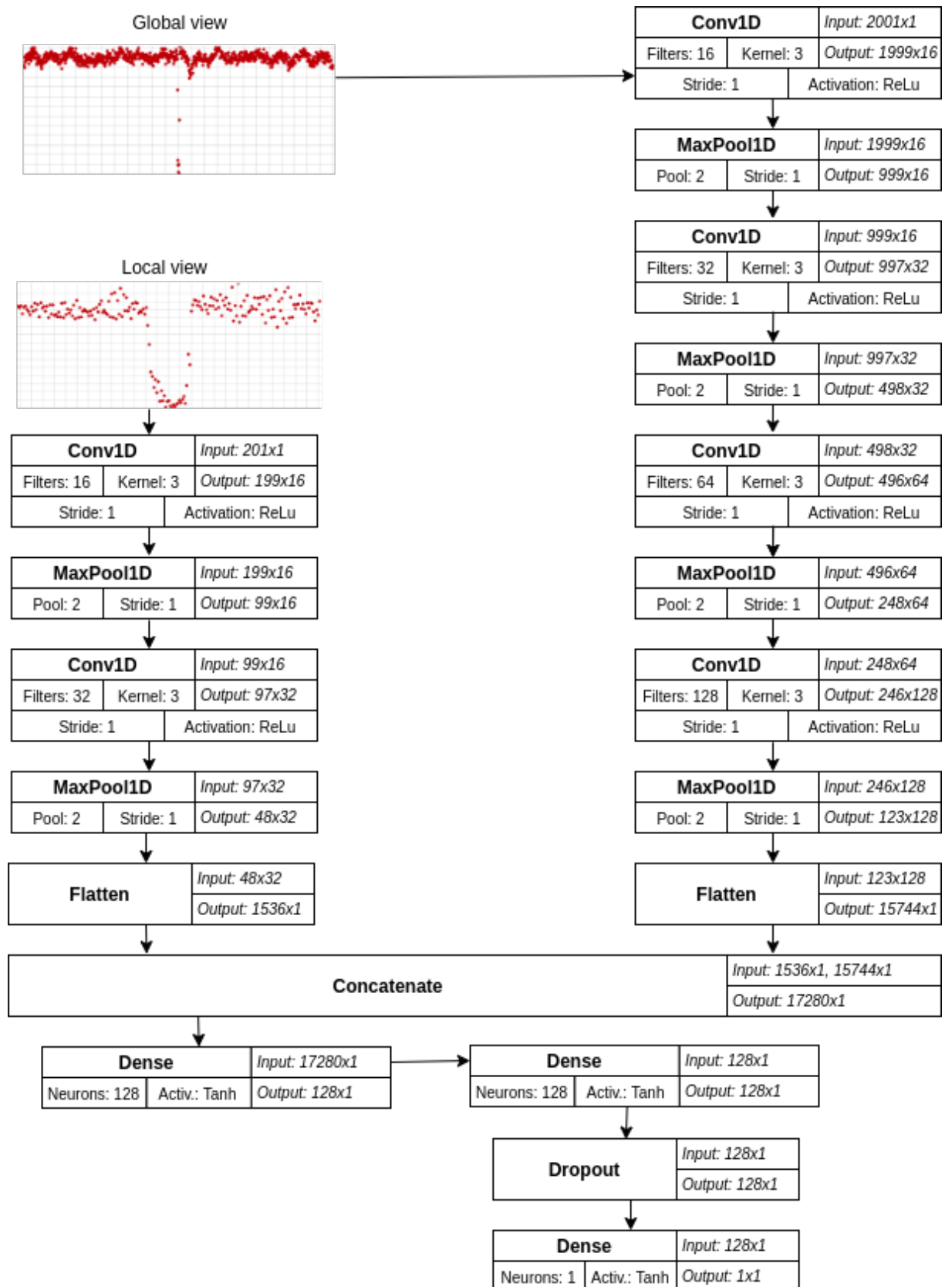
- [9] NORDEEN, Alex. MongoDB: Learn in 24 hours.. *Guru99*. 2020. [cit. 20. 2. 2021]. ISBN 1230004303503
- [10] OGBUEFI, Calvin. Photometry Analysis of Exoplanets WASP-80b & HD 189733b. *Baylor University*. [online]. 2013. [cit. 23. 10. 2020]. Dostupné z: <https://www.baylor.edu/content/services/document.php/208057.pdf>
- [11] PECINOVSKÝ, Rudolf. Python: kompletní příručka jazyka pro verzi 3.9.. *Praha: Grada Publishing*. 2020. [cit. 20. 2. 2021]. ISBN 978-80-271-1269-2
- [12] PERRYMAN, Michael. Extra-solar planets. *Reports on Progress in Physics 63(8)*. [online]. 31. 5. 2000. [cit. 8. 10. 2020]. Dostupné z: <https://arxiv.org/abs/astro-ph/0005602>
- [13] RUSELL, Stuart J., NORVIG, Peter. Artificial Intelligence: A Modern Approach (3rd ed.). *Upper Saddle River, New Jersey: Prentice Hall*. 2009. [cit. 29. 6. 2021]. ISBN 978-0-13-604259-4
- [14] SHALLUE, Christopher, VANDERBURG, Andrew. Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90. *The Astronomical Journal 155(2)*. [online]. 30. 1. 2018. [cit. 8. 10. 2020]. Dostupné z: <https://iopscience.iop.org/article/10.3847/1538-3881/aa9e09>
- [15] TASKER, Elizabeth, LANEUVILLE, Matthieu, GUTTENBERG, Nicholas. Estimating Planetary Mass with Deep Learning. *The Astronomical Journal 159(2)*. [online]. 25. 11. 2019. [cit. 9. 10. 2020]. Dostupné z: <https://arxiv.org/abs/1911.11035>
- [16] A Community Python Library for Astronomy. *astropy*. [online]. 2020. [cit. 21. 2. 2021]. Dostupné z: <https://docs.astropy.org/en/stable>
- [17] A friendly Python package for making discoveries with Kepler & TESS. *Lightkurve*. [online]. 2020. [cit. 21. 2. 2021]. Dostupné z: <https://docs.lightkurve.org>
- [18] Documentation. *styled components*. [online]. 2020. [cit. 21. 2. 2021]. Dostupné z: <https://styled-components.com/docs>

- [19] Getting started. *React*. [online]. 2021. [cit. 21. 2. 2020]. Dostupné z: <https://reactjs.org/docs/getting-started.html>
- [20] Introduction. *Socket.IO*. [online]. 2020. [cit. 21. 2. 2021]. Dostupné z: <https://socket.io/docs/v3>
- [21] Keras API reference. *Keras*. [online]. 2020. [cit. 29. 10. 2020]. Dostupné z: <https://keras.io/api>
- [22] Metody objevování planet. *Astronomia*. [online]. 23. 1. 2013. [cit. 23. 12. 2019]. Dostupné z: <http://hvezdy.astro.cz/exoplanety/51-metody-objevovani-planet>
- [23] mongoDB manual. *mongoDB*. [online]. 2021. [cit. 20. 2. 2020]. Dostupné z: <https://docs.mongodb.com/manual/reference/database-references>
- [24] MongoEngine User Documentation. *MongoEngine*. [online]. 2020. [cit. 21. 2.2021]. Dostupné z: <http://docs.mongoengine.org>
- [25] NASA Exoplanet Archive. *NASA Exoplanet Science Insititute*. [online]. 12. 8. 2019. [cit. 25. 12. 2019]. Dostupné z: [https://exoplanetarchive.ipac.caltech.edu/docs/API\\_exoplanet\\_columns.html](https://exoplanetarchive.ipac.caltech.edu/docs/API_exoplanet_columns.html)
- [26] Otevřená data NASA. *NASA*. [online]. 2020. [cit. 20. 2. 2020]. Dostupné z: <https://nasa.github.io/data-nasa-gov-frontpage>
- [27] TESS Exoplanet Mission. *NASA*. [online]. 24. 8. 2020. [cit. 8. 10. 2020]. Dostupné z: <https://www.nasa.gov/content/about-tess>
- [28] TypeScript Documentation. *TypeScript*. [online]. 2020. [cit. 21. 2. 2021]. Dostupné z: <https://www.typescriptlang.org/docs>
- [29] web development, one drop at a time. *Flask*. [online]. 2020. [cit. 21. 2. 2021]. Dostupné z: <https://flask.palletsprojects.com>

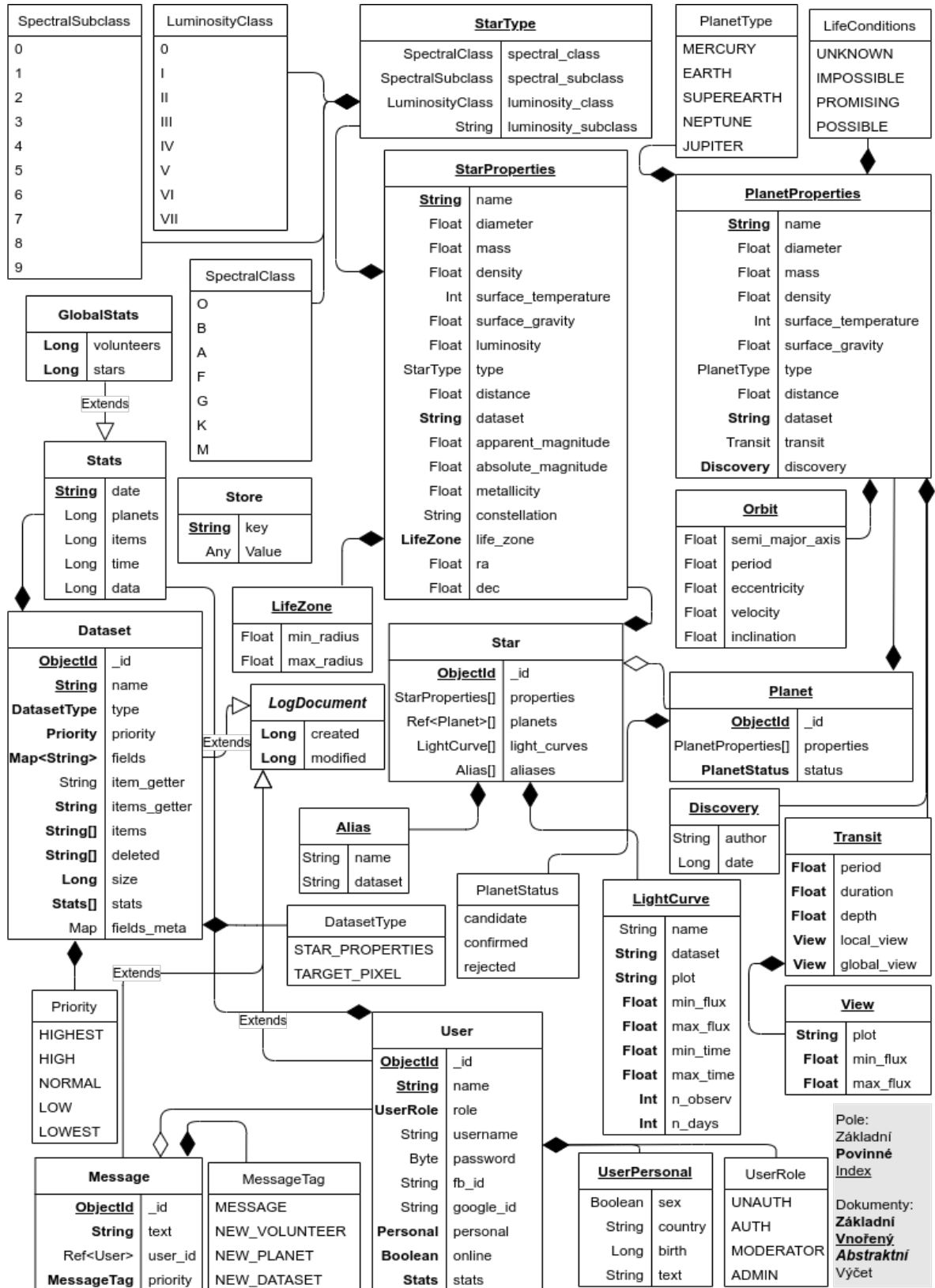
## SEZNAM PŘÍLOH

Příloha A	Architektura neuronové sítě .....	84
Příloha B	Architektura databáze .....	85
Příloha C	Dokumentace REST API .....	86
Příloha D	Verifikace výsledků .....	87

# PŘÍLOHA A – ARCHITEKTURA NEURON. SÍTĚ



# PŘÍLOHA B – ARCHITEKTURA DATABÁZE



# PŘÍLOHA C – DOKUMENTACE REST API

The screenshot displays a REST API documentation interface with the following endpoints:

- GET** /planets
- POST** /planets
- GET** /planets/ranks
- DELETE** /planets/{id}
- PUT** /planets/{id}
- GET** /planets/{id} (Detailed view)

The detailed view for the **GET /planets/{id}** endpoint includes the following sections:

- Parameters:** A table with columns 'Name' and 'Description'. It lists a required parameter 'id' of type 'string (path)' with a text input field containing 'id'. A 'Try it out' button is located to the right.
- Responses:** A dropdown menu for 'Response content type' is set to 'application/json'.
- Code:** A table with columns 'Code' and 'Description'. It lists a 200 status code with the description 'Successfully get Planet.' and a 404 status code with the description 'Planet with specified ID was not found.' Each entry includes an 'Example Value' and a 'Model' link.

The example JSON response for the 200 status code is as follows:

```
{
  "_id": "string",
  "properties": {
    "name": "string",
    "type": "mercury",
    "diameter": 0,
    "mass": 0,
    "density": 0,
    "orbital_period": 0,
    "orbital_velocity": 0,
    "surface_temperature": 0,
    "surface_gravity": 0,
    "semi_major_axis": 0,
    "life_conditions": "string",
    "distance": 0,
    "dataset": "string",
    "processed": true,
    "transit": {
      "period": 0,
      "depth": 0,
      "duration": 0,
      "local view": {
        "plot": "string",
        "min_flux": 0,
        "max_flux": 0
      },
      "global view": {
```

The example JSON response for the 404 status code is as follows:

```
{
  "type": "NOT_FOUND",
```

## PŘÍLOHA D – VERIFIKACE VÝSLEDKŮ

Pozitivní											
Planeta	Detekce	Perioda oběhu [d]			Rov. poloměr [ $r_{\oplus}$ ]			Hmotnost [ $m_{\oplus}$ ]			
		Vlastní	NASA	Chyba	Vlastní	NASA	Chyba	Vlastní	NASA	Chyba	
Kepler-39 b	Ano	21.087	21.087	0	14.8	14.77	0.0020	447	5688	0.9214	
Kepler-40 b	Ano	6.873	6.873	0	13.802	13.8	0.0001	379.5	674	0.4369	
Kepler-41 b	Ano	1.856	1.856	0	8.609	10.24	0.1593	136.9	174.8	0.2168	
Kepler-43 b	Ano	3.024	3.024	0	12.21	12.5	0.023	282.3	998	0.7171	
Kepler-45 b	Ano	2.455	2.455	0	10.853	11.03	0.0163	212.17	150.5	0.4147	
Kepler-63 b	Ano	9.434	9.434	0	5.64	6.11	0.0777	203	120	0.6917	
Kepler-66 b	Ano	17.816	17.816	0	2.389	2.81	0.1498	14.16			
Kepler-69 b	Ano	13.722	13.722	0	2.165	2.28	0.0504	10.43			
Kepler-69 c	Ne		242.467			1.74					
Kepler-74 b	Ano	7.341	7.341	0	14.1	14.31	0.014	396	216	0.8333	
Kepler-75 b	Ano	8.887	8.887	0	9.48	10.82	0.1238	167	3146	0.9440	
Kepler-76 b	Ano	1.545	1.545	0	9.74	13.67	0.2875	176	636	0.7232	
Kepler-77 b	Ano	3.579	3.579	0	10.719	10.761	0.0039	206	225	0.0844	
Kepler-78 b	Ano	0.355	0.355	0	1.315	1.2	0.0958	2.85	1.96	0.4541	
Kepler-79 b	Ne		13.485		3.47			10.9			
Kepler-79 c	Ne		27.402		3.72			12.5			
Kepler-79 d	Ano	52.09	52.09	0	7.244	7.239	0.0007	96.3			
Kepler-80 b	Ano	7.053	7.053	0	2.07	2.21	0.0676	9.1	6.94	0.3112	
Kepler-91 b	Ano	6.25	6.25	0	13.9	14.5	0.0414	385	241	0.5975	
Kepler-95 b	Ano	11.523	11.523	0	3.22	3.13	0.0288	35.6			
Kepler-96 b	Ano	16.238	16.238	0	2.55	2.55	0	17.4	8.46	1.0567	
Kepler-98 b	Ano	1.542	1.542	0	1.59	1.82	0.1264	3.99	3.55	0.1103	
Kepler-99 b	Ano	4.604	4.604	0	1.48	1.48	0	3.21	1.88	0.7074	
Kepler-425 b	Ano	3.797	3.797	0	9.48	9.84	0.0366	167	79.45	1.1019	
Kepler-426 b	Ano	3.218	3.218	0	9.52	12.22	0.2209	169	108	0.5648	
Negativní											
KIC	Detekce	KIC	Detekce	KIC	Detekce	KIC	Detekce	KIC	Detekce	KIC	Detekce
100000925	Ne	100000926	Ne	100000927	Ne	100000928	Ne	100000929	Ne		
100000930	Ne	100000931	Ne	100000932	Ne	100000933	Ne	100000934	Ne		
100000935	Ne	100000936	Ne	100000937	Ne	100000938	Ne	100000939	Ne		
100000940	Ne	100000941	Ne	100000942	Ne	100000943	Ne	100000944	Ne		
100000945	Ne	100000946	Ne	100000947	Ne	100000948	Ne	100000949	Ne		
100000950	Ne	100000951	Ne	100000952	Ne	100000953	Ne	100000954	Ne		
100000955	Ne	100000956	Ne	100000957	Ne	100000958	Ne	100000959	Ne		
100000960	Ne	100000961	Ne	100000962	Ne	100000963	Ne	100000964	Ne		
4928826	Ne	4928862	Ne	4928898	Ne	4932320	Ne	4936238	Ne		
4936281	Ne	4936287	Ne	4940980	Ne	4940999	Ne	4941156	Ne		
4929214	Ne	4929259	Ne	4929301	Ne	4929426	Ne	4929449	Ne		
8396995	Ne	8397017	Ne	8397026	Ne	8397068	Ne	8397111	Ne		