

UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2021

Michal Struna

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Detekce a analýza exoplanet s využitím
distribuovaných výpočtů a umělé inteligence

Michal Struna

Diplomová práce
2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚleckého díla, Uměleckého výkonu)

Jméno a příjmení: **Michal Struna**

Osobní číslo: **I16144**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Webový 3D simulátor těles ve vesmíru**

Zadávající katedra: **Katedra informačních technologií**

Zá s a d y p r o v y p r a c o v á n í :

Práce se zabývá tvorbou webové typescriptové aplikace pro 3D vizualizaci těles ve vesmíru. V rámci práce je kláden důraz na dynamický obsah, na kterém se mohou všichni uživatelé po úspěšné autentifikaci podílet. Data jsou ukládána do databáze na serveru. Pro práci s databází je využito REST API. Cílem bakalářské práce je vytvořit webovou aplikaci v jazyce TypeScript, jejímž obsahem je 3D simulátor těles ve vesmíru v reálném čase. Aplikace se skládá z klientské a serverové části. Klientská část zahrnuje: - Uživatelské rozhraní v Reactu a ostylované v SASS umožňující uživatelům autentifikaci, zobrazení a úpravu obsahu a písemnou komunikaci s ostatními uživateli. - 3D simulátor využívající knihovnu THREE.js zobrazující tělesa v reálném čase. Průběh času je možné zrychlovat, zpomalovat či vracet. Serverová část obsahuje: - Serverová aplikace napsaná v Node.js poskytující REST API pro práci s daty. - Data budou uložena v MongoDB databázi, se kterou se bude pracovat za využití knihovny Mongoose. - Dokumentace REST API pomocí nástroje Swagger. Výstupem práce je aplikace, kterou je po nainstalování závislostí pomocí balíčkovacího systému npm a transpilaci TypeScriptu do JavaScriptu možné okamžitě spustit.

Rozsah grafických prací:

Rozsah pracovní zprávy: min. 30 s., dop. rozsah 40 s.

Forma zpracování bakalářské práce: tištěná

Seznam odborné literatury:

- KLECZEK, Josip. Velká encyklopédie vesmíru. Praha: Academia, 2002s., 48s.
barev. obr. příl. ISBN 80-200-0906-x
- REES, Martin J, ed. Vesmír: [obrazová encyklopédie]. Přeložil Pavel
PŘÍHODA. V Praze: Knižní klub, 2006. ISBN 80-242-1668-x
- JPL Solar System Dynamics. JPL Solar System Dynamics [online]. Dostupné z:
<https://ssd.jpl.nasa.gov>
- MARDAN, Azat. Practical Node.js: building real-world scalable web apps.
Berkeley, California: Apress, [2014]. Expert's voice in Web development. ISBN
978-1-4302-6595-5
- MARDAN, Azat. Practical Node.js: building real-world scalable web apps.
Berkeley, California: Apress, [2014]. Expert's voice in Web development. ISBN
978-1-4302-6595-5

Vedoucí bakalářské práce:

Ing. Monika Borkovcová, Ph.D.

Katedra informačních technologií

Datum zadání bakalářské práce: 31. října 2018

Termín odevzdání bakalářské práce: 12. května 2019


Ing. Zdeněk Němec, Ph.D.
děkan




Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

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

...

ANOTACE

...

KLÍČOVÁ SLOVA

exoplanety, extrasolární planety, kepler, umělá inteligence, python

TITLE

Artificial intelligence for exoplanet detection from transit data

ANNOTATION

...

KEYWORDS

exoplanet, extrasolar planets, kepler, artificial intelligence, python

OBSAH

Seznam obrázků	10
Seznam tabulek	12
Seznam zdrojových kódů	13
Seznam vzorců	14
Seznam obrázků	15
Úvod	16
1 Hledání exoplanet	17
1.1 Tranzitní metoda	20
1.1.1 Target pixel file	21
1.1.2 Světelná křivka hvězdy	21
1.1.3 Vyřazení false positives	23
1.1.4 Výpočet vlastností planety	24
1.1.5 Výpočet hmotnosti planety	25
1.2 Metoda radiálních rychlostí	26
1.2.1 Výpočet radiální rychlosti hvězdy	27
1.2.2 Výpočet hmotnosti planety	27
1.3 Astrometrická metoda	28
1.4 Přímé zobrazení	29
1.5 Gravitační mikročočky	29
1.6 Časování pulsarů	30
1.7 Časování tranzitů	32
1.7.1 Aplikace genetického algoritmu	33
2 Umělé neuronové sítě	34
2.1 Neuron	34
2.1.1 Vstupy neuronu	35
2.1.2 Váhy spojení	35

2.1.3	Práh neuronu	36
2.1.4	Agregační funkce	36
2.1.5	Aktivační funkce	36
2.2	Typy neuronových sítí	37
2.2.1	Jednoduchý perceptron	37
2.2.2	Dopředná vícevrstvá umělá neuronová síť	38
2.2.3	Konvoluční neuronová síť	39
2.3	Typy vrstev neuronových sítí	41
2.3.1	Plně propojená vrstva	41
2.3.2	Konvoluční vrstva	41
2.3.3	Vrstva Max-Pooling	43
2.3.4	Vrstva Flatten	44
2.3.5	Vrstva Dropout	44
2.4	Učení umělé neuronové sítě	45
2.4.1	Algoritmus zpětného šíření chyby	45
3	Další oblasti umělé inteligence	47
3.1	Genetický algoritmus	48
3.1.1	Inicializace	49
3.1.2	Selekce	50
3.1.3	Křížení	50
3.1.4	Mutace	51
4	Projekty pro hledání exoplanet	52
4.1	Planet Hunters	52
4.2	Astronet	52
5	Použité technologie	53
5.1	TypeScript	53
5.1.1	React	53
5.1.2	Styled components	53
5.2	Python	54
5.2.1	Flask	54
5.2.2	Astropy	54

5.2.3	LightKurve	55
5.2.4	Keras	55
5.2.5	MongoEngine	55
5.3	MongoDB	56
5.4	Socket.io	56
6	Návrh a vývoj aplikace	57
6.1	Server	57
6.1.1	REST API	58
6.1.2	Socket.io API	59
6.2	Webová aplikace	59
6.3	Klientská aplikace	60
6.4	Neuronová síť	60
6.5	Databáze	60
6.6	Datasetsy	60
6.6.1	Target pixel files	61
6.6.2	Světelné křivky hvězd	61
6.6.3	Radiální rychlosti hvězd	61
6.6.4	Hvězdy	61
6.6.5	Planety	62
6.6.6	Názvy	63
7	Rozvržení aplikace	64
7.1	Přehled	64
7.2	Databáze	64
7.3	Detail systému	64
7.4	Objevování	64
7.5	Nápověda	64
7.6	Autentizace	64
Závěr		65
Použitá literatura		65
Seznam příloh		68

SEZNAM OBRÁZKŮ

Obrázek 1	Četnost objevů exoplanet v jednotlivých letech	17
Obrázek 2	Počty objevených exoplanet jednotlivými metodami	18
Obrázek 3	Známé exoplanety dle jejich vlastností a metody objevení	18
Obrázek 4	Počty objevených exoplanet dle jejich typu	19
Obrázek 5	Přechod planety přes kotouč hvězdy	20
Obrázek 6	Target pixel file hvězdy Kepler-10	21
Obrázek 7	Světelná křivka soustavy Kepler-13	22
Obrázek 8	Dvojhvězda KIC 8262223	23
Obrázek 9	Cefeida KIC 3733346	23
Obrázek 10	Proměnná hvězda KIC 9832227	23
Obrázek 11	Kataklizmická proměnná hvězda KIC 9406652	23
Obrázek 12	Světelná křivka Kepler-10	23
Obrázek 13	Složená světelná křivka	23
Obrázek 14	Globální pohled na tranzit Kepler-10 c	24
Obrázek 15	Lokální pohled na tranzit Kepler-10 c	24
Obrázek 16	Metoda radiálních vzdáleností	26
Obrázek 17	Radiální rychlosť hvězdy 51 Pegasi v čase	27
Obrázek 18	Obíhání hvězdy a planety kolem společného těžiště	28
Obrázek 19	Kolísání hvězdy Gliese 876 s planetou	28
Obrázek 20	Hvězda GQ Lupi s přímo pozorovanou exoplanetou	29
Obrázek 21	Princip gravitační čočky	29
Obrázek 22	Ilustrace gravitační mikročočky OGLE 2003-BLG-235	30
Obrázek 23	Ilustrace pulsaru s obíhající planetou	31
Obrázek 24	Nepravidelnosti v tranzitech planety Kepler-46b	32
Obrázek 25	Princip časování tranzitů	33
Obrázek 26	Model formálního neuronu	35
Obrázek 27	Příklady aktivačních funkcí neuronu	37
Obrázek 28	Lineárně separovatelná (vlevo) a neseparovatelná (vpravo) úloha	37
Obrázek 29	Zapojení neuronů	38
Obrázek 30	Architektura dopředné vícevrstvé umělé neuronové sítě	38

Obrázek 31	Ručně psané číslice 9	39
Obrázek 32	Vztah mezi počítačovým viděním, strojovým učením a konvoluční sítí	39
Obrázek 33	Příklad konvoluční sítě pro rozpoznávání ručně psaných číslic	40
Obrázek 34	Postupné posouvání konvolučního filtru přes vstup	41
Obrázek 35	Příklad aplikace konvolučního filtru na vstup	42
Obrázek 36	Příklad operace Max-Pooling	43
Obrázek 37	Příklad fungování vrstvy Flatten	44
Obrázek 38	Příklad fungování vrstvy Dropout	44
Obrázek 39	Příklad algoritmu zpětného šíření chyby	46
Obrázek 40	Rozdělení umělé inteligence	48
Obrázek 41	Některí z vygenerovaných jedinců GA pro TSP	49
Obrázek 42	Příklad křížení jedinců GA pro TSP	50
Obrázek 43	Příklad mutace jedince GA pro TSP	51
Obrázek 44	Hledání řešení TSP pomocí GA	51
Obrázek 45	Komponenty projektu a jejich komunikace	57
Obrázek 46	Architektura serveru	57
Obrázek 47	Architektura webové aplikace	59
Obrázek 48	Stavy procesů a přechody mezi nimi	60
Obrázek 49	Trénovací množina	60
Obrázek 50	Část datasetu s vlastnostmi hvězd	61
Obrázek 51	Část datasetu s vlastnostmi planet	62

SEZNAM TABULEK

Tabulka 1	Známé typy exoplanet	19
Tabulka 2	Fáze oběhu tranzitující exoplanety	22
Tabulka 3	Veličiny tranzitu	22
Tabulka 4	Příklady výpočtu hmotnosti planet	28
Tabulka 5	Komponenty formálního neuronu	34
Tabulka 6	Parametry plně propojené vrstvy	41
Tabulka 7	Parametry konvoluční vrstvy	42
Tabulka 8	Parametry vrstvy Max-Pooling	43
Tabulka 9	Parametry vrstvy Dropout	44
Tabulka 10	Údaje o hvězdách ukládané do databáze	62
Tabulka 11	Údaje o planetách ukládané do databáze	62
Tabulka 12	Pojmenování soustavy Kepler-10 v různých katalogích	63

SEZNAM ZDROJOVÝCH KÓDŮ

Zdrojový kód 1	Pseudokód genetického algoritmu	49
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

SEZNAM VZORCŮ

1	Pravděpodobnost zpozorování tranzitu planety přes hvězdu	21
2	Výpočet velké poloosy dráhy planety	24
3	Výpočet poloměru planety	25
4	Odhad průměrné rychlosti oběhu planety	25
5	Výpočet inklinace dráhy	25
6	Radiální rychlosť na základě změny vlnové délky	27
7	Radiální rychlosť	27
8	Poměr zářivého výkonu hvězdy a planety	29
9	Funkce vyjadřující přesnost modelu vůči realitě při počítání TTV	32
10	Rovnice problému 3 těles	33
11	Obecný vztah vstupu neuronu a jeho váhy	35
12	Lineární vztah vstupu neuronu a jeho váhy	36
13	Agregační funkce neuronu	36
14	Hyperbolicko-tangenciální aktivační funkce	36
15	Sigmoidální aktivační funkce	36
16	Gaussova aktivační funkce	36
17	Provádění konvoluce pro výřez vstupu a filtr	42
18	Výpočet chyby pro jeden vzor u algoritmu zpětného šíření chyby	45
19	Přírustek váhy u algoritmu zpětného šíření chyby	45
20	Výpočet lokálního gradientu neuronu	46

SEZNAM ZKRATEK

au	Astronomical unit
CNN	Convolutional neural network
csv	Comma-separated values
FC	Fully-connected layer
FFNN	Feed-forward neural network
fits	Flexible image transport system
ly	Light year
ODM	Object-document mapper
pc	Parsec
TTV	Transit timing variation

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. [10]

První potvrzená exoplaňta 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. [11, 14]

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 Keplera – satelit TESS – od něhož je očekáván objev 20 000 exoplanet. [16]

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 takového pozorování jsou často stovky GiB fyzikálních a statistických dat, jež je následně nutno zpracovat. [10]

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.

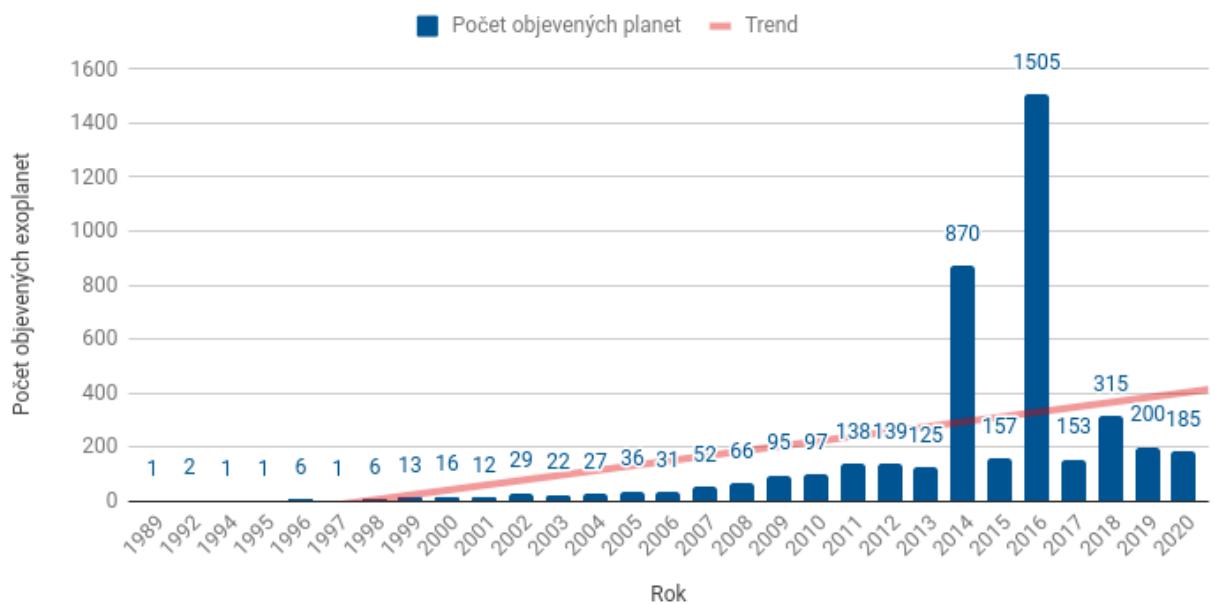
Díky distribuovaným výpočtům se do výzkumu exoplanet bude moci bez vysokého úsilí, znalosti či technického vybavení zapojit i široká veřejnost. To může urychlit vývoj a zároveň zvýšit povědomí o této vědní disciplíně.

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 vykovádající 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 se omezujeme 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. Můžeme se tak přiblížit odpovědím na otázky jako „Jak vzácné jsou podmínky pro život ve vesmíru?“ nebo „Jak vznikla a jak se vyvíjela naše planeta?“ [10]

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. [10, 14]

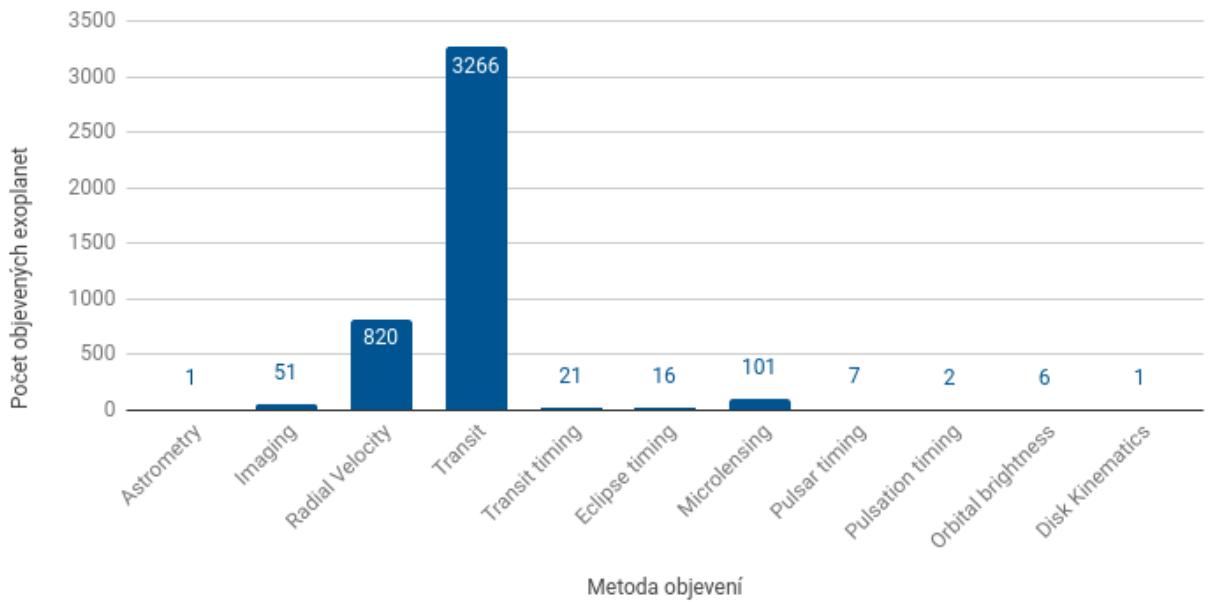


Obrázek 1: Četnost objevů exoplanet v jednotlivých letech ¹

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

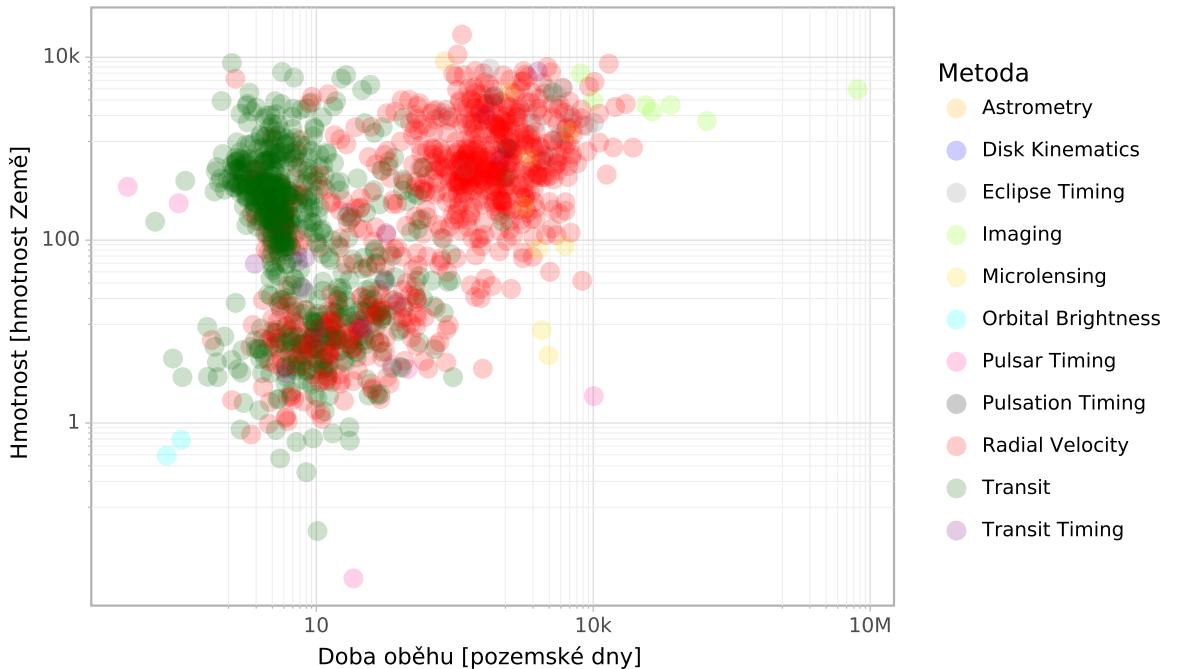
Jednotlivé metody budou popsány v následujících podkapitolách. Zdaleka nejvýznamnější je tranzitní metoda, kterou byla objevena většina exoplanet. Velké množství planet bylo objeveno také metodou radiálních rychlostí. [14]

¹ Vytvořeno autorem, zdroj dat: [14].



Obrázek 2: Počty objevených exoplanet jednotlivými metodami ¹

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. [14]



Obrázek 3: Známé exoplanety dle jejich vlastností a metody objevení ¹

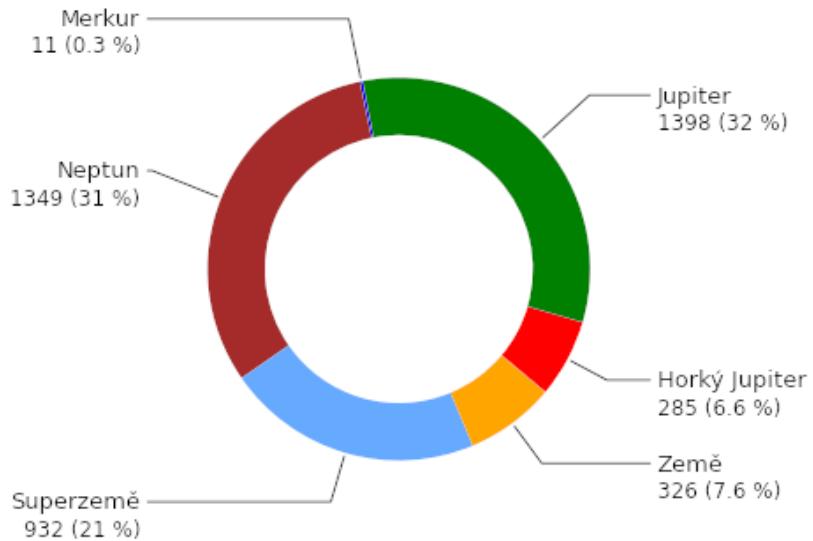
¹ Vytvořeno autorem, zdroj dat: [14].

Na základě známých charakteristik můžeme 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 detektovat.	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 helium 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.	Neexistuje
Horký Jupiter	Zvláštní typ plynných obrů, které narozdíl od těch v naší soustavě obíhají v těsné blízkosti své hvězdy. Jsou rozptálené na tisíce K a mohou významně působit na svou hvězdu. Díky tomu jsou první objevené exoplanety právě horké Jupitery.	Neexistuje

Tabulka 1: Známé typy exoplanet ¹

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

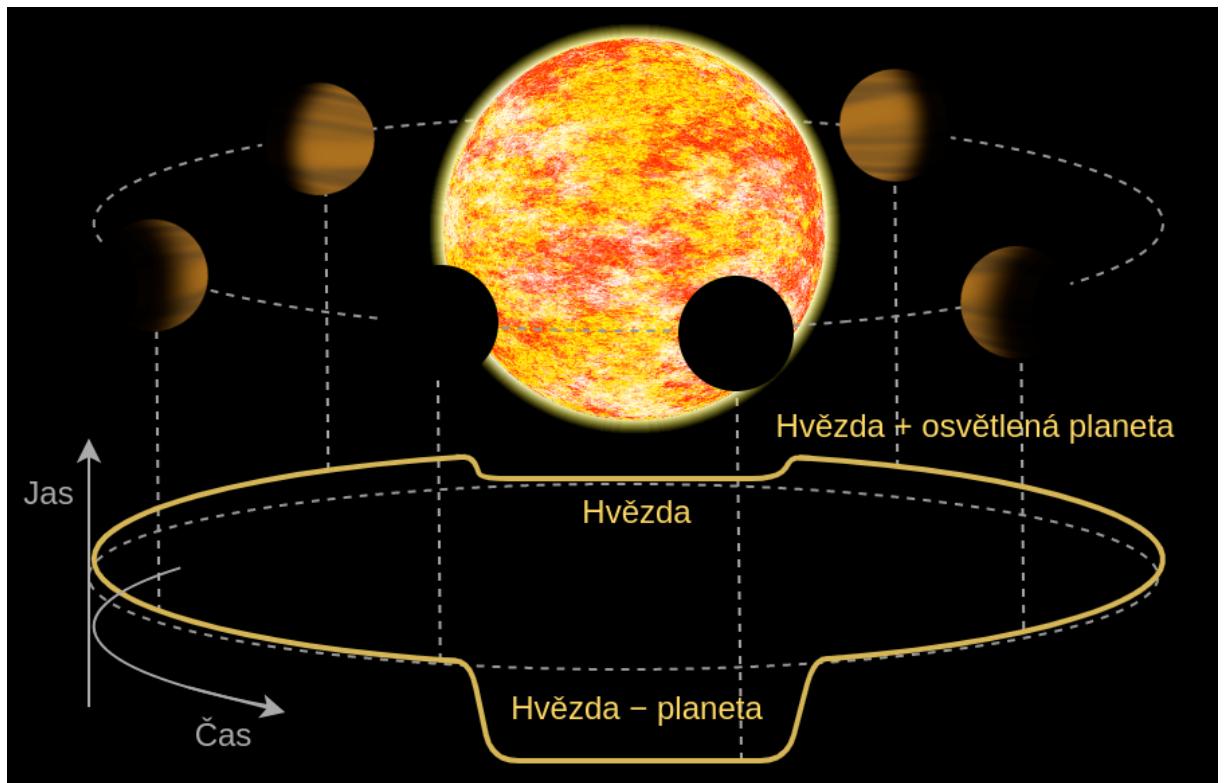


Obrázek 4: Počty objevených exoplanet dle jejich typu ¹

¹ Vytvořeno autorem, zdroj dat: [14].

1.1 Tranzitní metoda

Někdy se planeta při obíhání dostane mezi svou hvězdu a Zemi. 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. [10, 8]



Obrázek 5: Přechod planety přes kotouč hvězdy ¹

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. [11]

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. [13, 8]

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

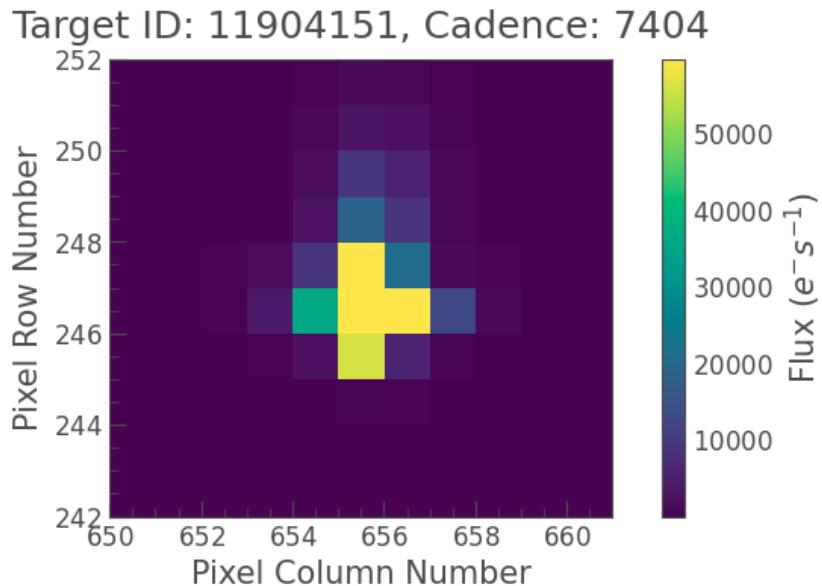
$$P = \frac{d_s}{a} \quad (1)$$

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 vyextrahuje 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.

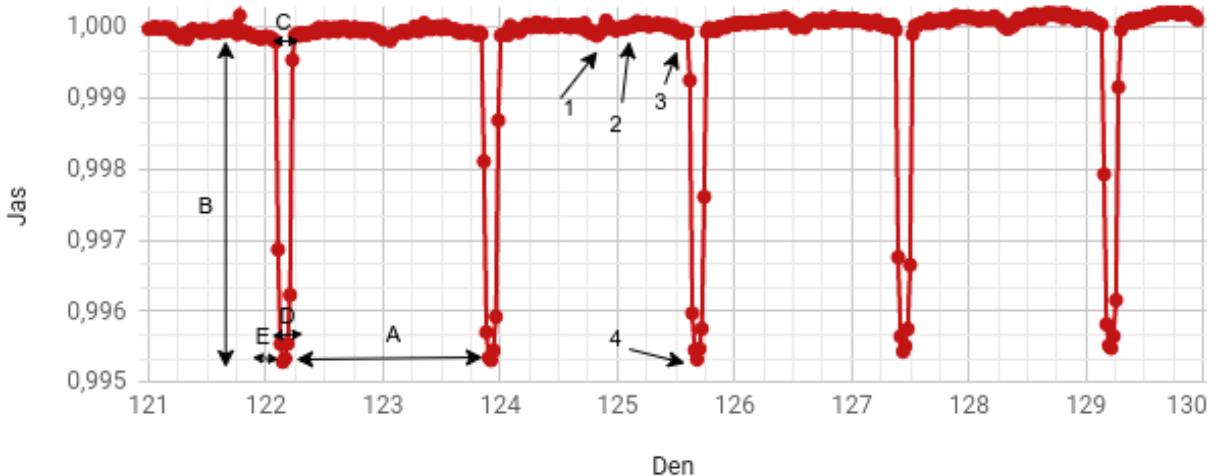


Obrázek 6: Target pixel file hvězdy Kepler-10 ¹

1.1.2 Světelná křivka hvězdy

Po složení všech TPF do časové řady a vypočítání jejich jasu dostaneme světelnou křivku. 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 detektovat planetu je obtížnější.

¹ Vytvořeno autorem, zdroj dat: [14].



Obrázek 7: Světelná křivka soustavy Kepler-13 ¹

Na obrázku 6 jsou čísla označeny jednotlivé fáze dále popsáné 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 + neosvětelná část planety
4 Planeta je před hvězdou (tranzit).	Hvězda – planeta

Tabulka 2: Fáze oběhu tranzitujející 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	

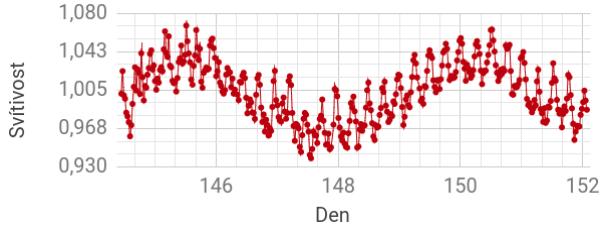
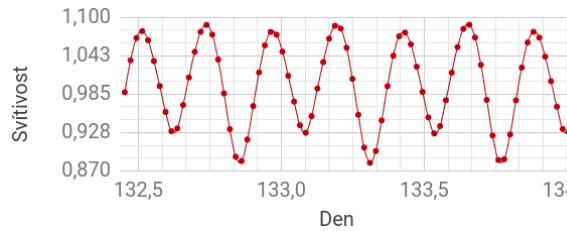
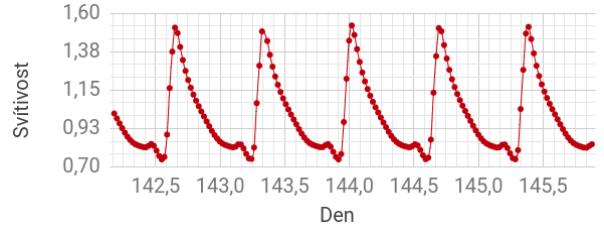
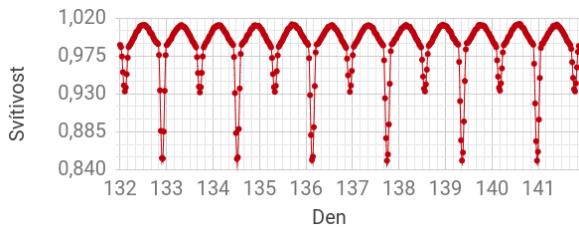
Tabulka 3: Veličiny tranzitu

TODO: Transit types image.

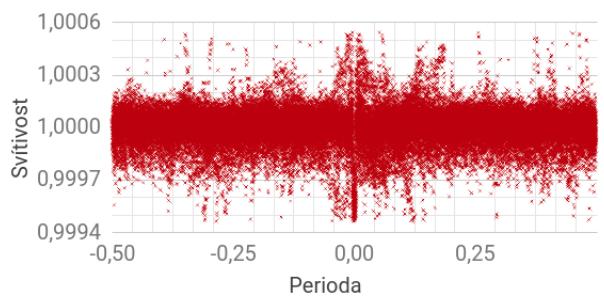
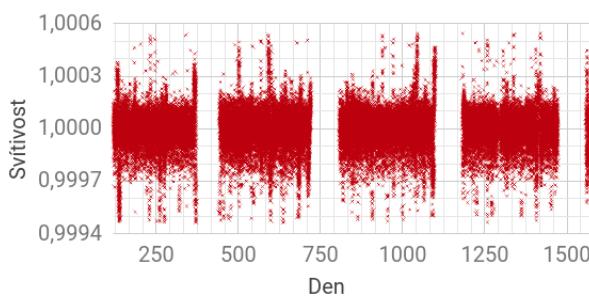
¹ Vytvořeno autorem, zdroj dat: [14].

1.1.3 Vyřazení false positives

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 popsaný v předchozí kapitole, je možné ho s určitou úspěšností rozpoznat pomocí naučené umělé neuronové sítě automaticky. [11]



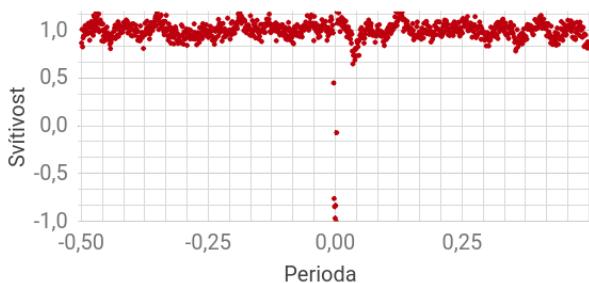
Je třeba aby vstupní data do neuronové sítě měla stejné rozměry i formát. Vzhledem k různorodosti světelných křivek je nutno provést několik kroků, abychom dosáhli standardizovaného formátu. Prvním krokem je složení časové řady do jedné periody, čímž dojde k posílení viditelnosti transitu (pokud zde nějaký je), nebo naopak k jeho vyrušení (pokud zde žádný není). [11]



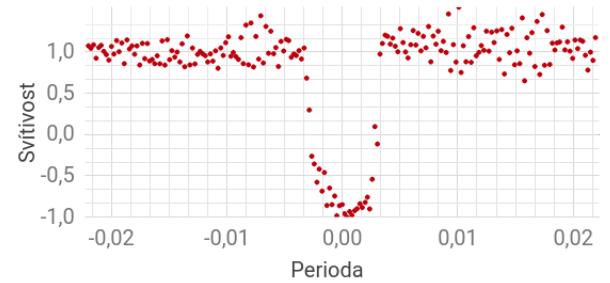
¹ Vytvořeno autorem, zdroj dat: [14].

Na obrázku 12 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. 13*) – Šířka periody a počet bodů v časové řadě jsou fixní. Nevýhodou je, že u planet s dlouhou periodou bude tranzit velice nepatrnný, proto pouze globální pohled nestačí. [11]
- **Lokální pohled** (*obr. 14*) – 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. [11]



Obrázek 14: Globální pohled na tranzit
Kepler-10 c¹



Obrázek 15: Lokální pohled na tranzit
Kepler-10 c¹

Fixní počet bodů lze zajistit nahrazením každých $\frac{\alpha}{\beta}$ sousedních bodů (α – současný počet bodů, β – požadovaný počet bodů) jediným, který bude reprezentovat jejich medián. Pro potřeby neuronové sítě je nutno oba pohledy také 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. [11]

1.1.4 Výpočet vlastností planety

Velkou poloosu dráhy planety lze vypočítat, pokud známe hmotnost hvězdy a periodu oběhu z třetího Keplerova zákona. [9]

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

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 lze vypočítat poloměr tranzitujeící planety po vyjádření z následující rovnice [8, 9]:

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

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

Vzorec 3: Výpočet poloměru planety

r = poloměr planety R = poloměr hvězdy F = jas hvězdy ΔF = změna jasu

Dále je možno odhadnout i průměrnou rychlosť pohybu planety po oběžné dráze. Skutečná průměrná rychlosť však může být jiná, protože se nepočítá s excentricitou dráhy [9]:

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

Vzorec 4: Odhad průměrné rychlosti oběhu planety

v = rychlosť oběhu planety a = velká poloosa dráhy planety T = perioda oběhu planety

Další z důležitých charakteristik orbity je inklinace (sklon), která nám řekne, jaký úhel svírají roviny pozorovatele a oběhu exoplanety. Bohužel lze zjistit minimální úhel sklonu dráhy a nikoliv jeho přesnou hodnotu. Inklinace se bude zpravidla blížit 90° . [9]

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

Vzorec 5: Výpočet inklinace dráhy

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

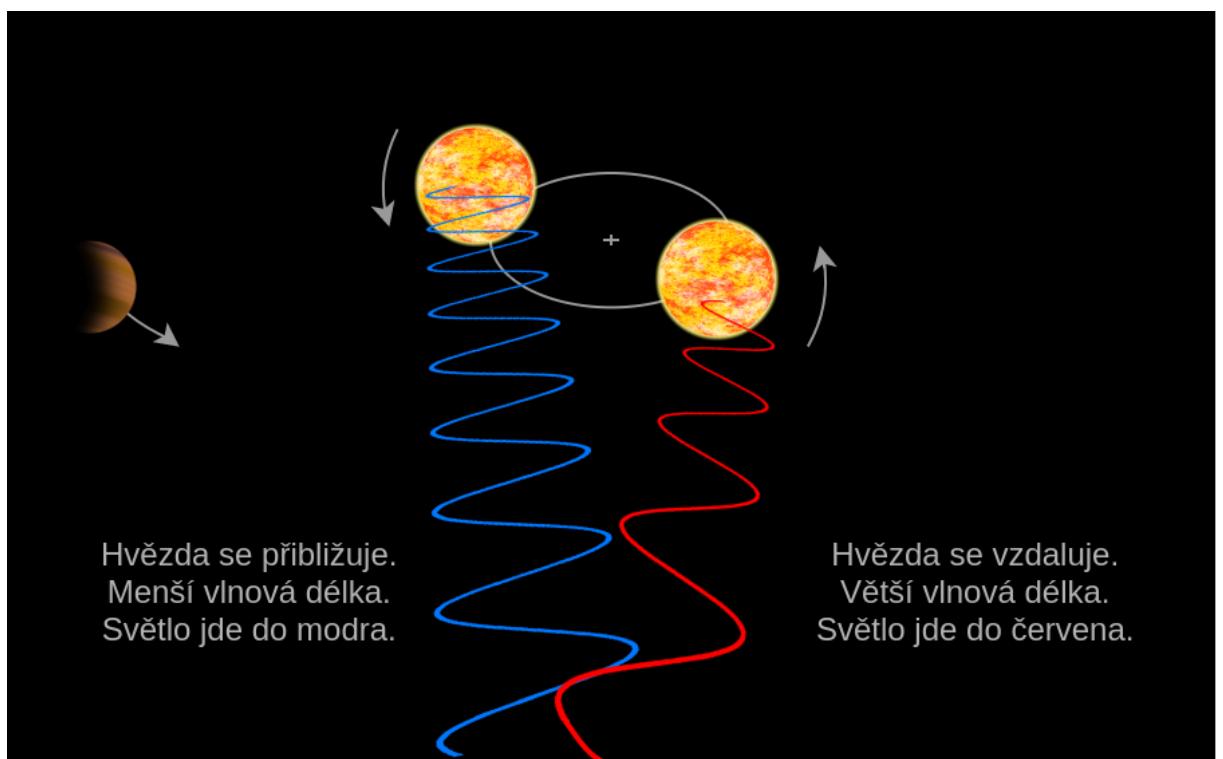
1.1.5 Výpočet hmotnosti planety

Hmotnost planety z tranzitní metody nelze vypočítat. Avšak z empirických dat je možné ji odhadnout na základě kombinace jiných veličin pomocí umělé neuronové sítě. TODO. [12]

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 vzdálování hvězdy vůči pozorovateli na Zemi. Právě pojem radiální rychlosť označuje rychlosť pohybu ve směru k pozorovateli. [13]

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 vzdálování se zdroje vlnění od pozorovatele. Vlnová délka se zvětšuje a barva jde do červena. Tomuto efektu se říká červený (resp. modrý) posuv. [13]



Obrázek 16: Metoda radiálních vzdáleností ¹

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 zpravidla vnímáme vyšším tónem, protože má menší vlnovou délku (vyšší frekvenci). Při vzdalování zdroje má zvuk větší vlnovou délku a je vnímán hlubším tónem. [6] Periodicky se opakující změny ve vlnové délce záření hvězdy tak mohou být důsledkem existence tělesa v této soustavě. [13]

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

1.2.1 Výpočet radiální rychlosti hvězdy

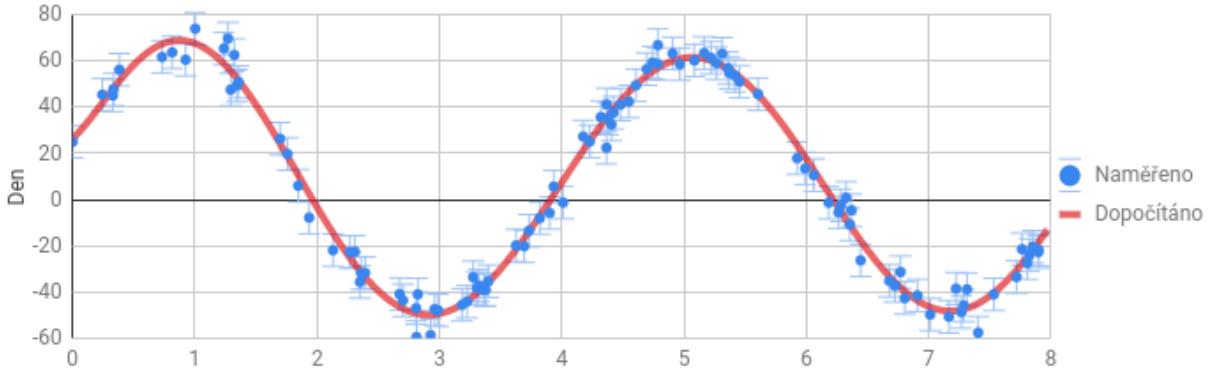
Poté, co teleskop sesbírá dostatečně velkou časovou řadu vlnových délek záření hvězdy může dojít k vypočítání radiální rychlosti v čase. Platí, že radiální rychlosť je kladná, pokud se zdroj od pozorovatele vzdaluje a záporná pokud se přibližuje. [13]

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

Vzorec 6: Radiální rychlosť na základě změny vlnové délky

$\Delta\lambda$ = změna vlnové délky λ_0 = klidová vlnová délka v = radiální rychlosť

Graf znázorňující radiální rychlosť hvězdy 51 Pegasi může vypadat takto:



Obrázek 17: Radiální rychlosť hvězdy 51 Pegasi v čase ¹

Na první pohled je patrná jedna periodická složka s periodou 4,23 dne a amplitudou 66,67 $\frac{m}{s}$, která značí, že kolem této hvězdy obíhá jedna planeta. Aby metoda radiálních rychlosť 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 rychlosťí 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. [13]

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

Vzorec 7: Radiální rychlosť

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

¹ Vytvořeno autorem, zdroj dat: [7].

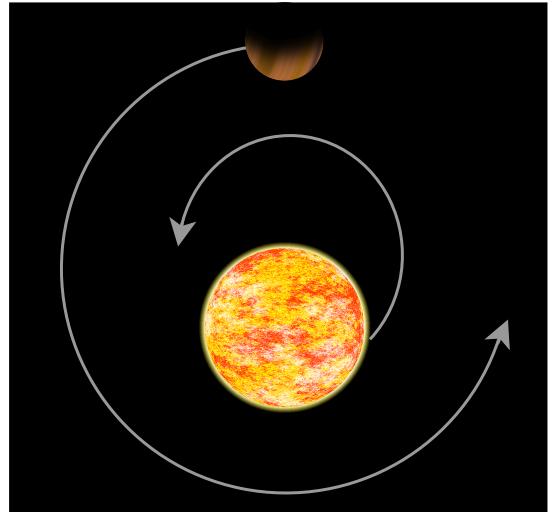
Těleso	Hvězda	Δv_{max} [m/s]	M_p [kg]	$\sin(i)$	e	T [r]	M_s [kg]
Země	Slunce	0,089	$2 * 10^{30}$	1	0,017	1	$5,97 * 10^{24}$
Jupiter		12,4			0,048	11,86	$1,9 * 10^{27}$
Pluto		0,00003			0,247	247,41	$1,3 * 10^{22}$
α Cen Bb	α Cen B	0,51	$1,8 * 10^{30}$		0	0,0089	$6,75 * 10^{24}$
51 Pegasi b	51 Pegasi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

Tabulka 4: Příklady výpočtu hmotnosti planet ¹

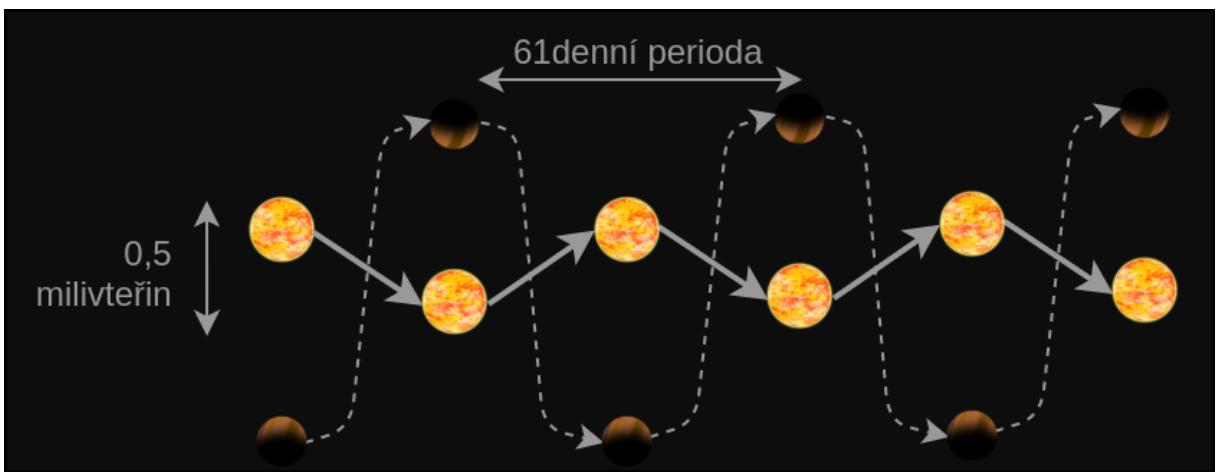
1.3 Astrometrická metoda

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. [13]

Pohyb hvězdy tak není přímočarý, ale vlnitý. Kolísání hvězdy je však pro pozorovatele na Zemi často pouze v řádu stovek úhlových mikrovteřin. [13] Z tohoto důvodu byla astrometrickou metodou dosud objevena pouze jediná exoplaneta. Dá se však očekávat, že se zlepšující se technikou bude tato metoda úspěšnější. [14]



Obrázek 18: Obíhání hvězdy a planety kolem společného těžiště ¹



Obrázek 19: Kolísání hvězdy Gliese 876 s planetou ²

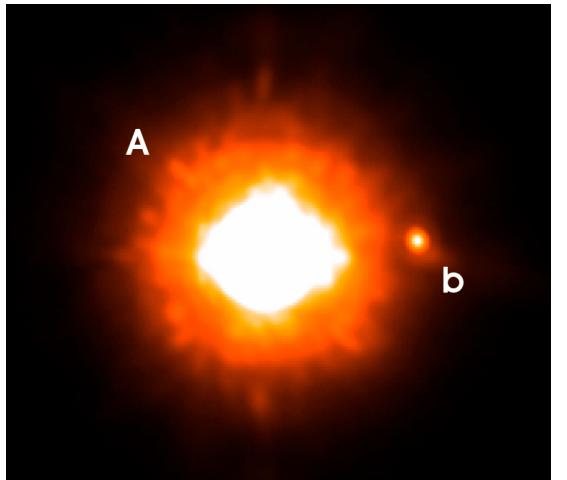
¹ Vytvořeno autorem, zdroj dat: [6, 7, 13, 14].

² 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} . [13]

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. [13]



Obrázek 20: Hvězda GQ Lupi s přímo pozorovanou exoplanetou¹

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

Vzorec 8: Poměr zářivého výkonu hvězdy a planety

α = odrazivost povrchu planety (albedo) | R_p = poloměr planety
 a = velká poloosa oběžné dráhy planety

1.5 Gravitační mikročočky

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 tak tuto přímou trajektorii vnímáme jako zakřivenou. [13]

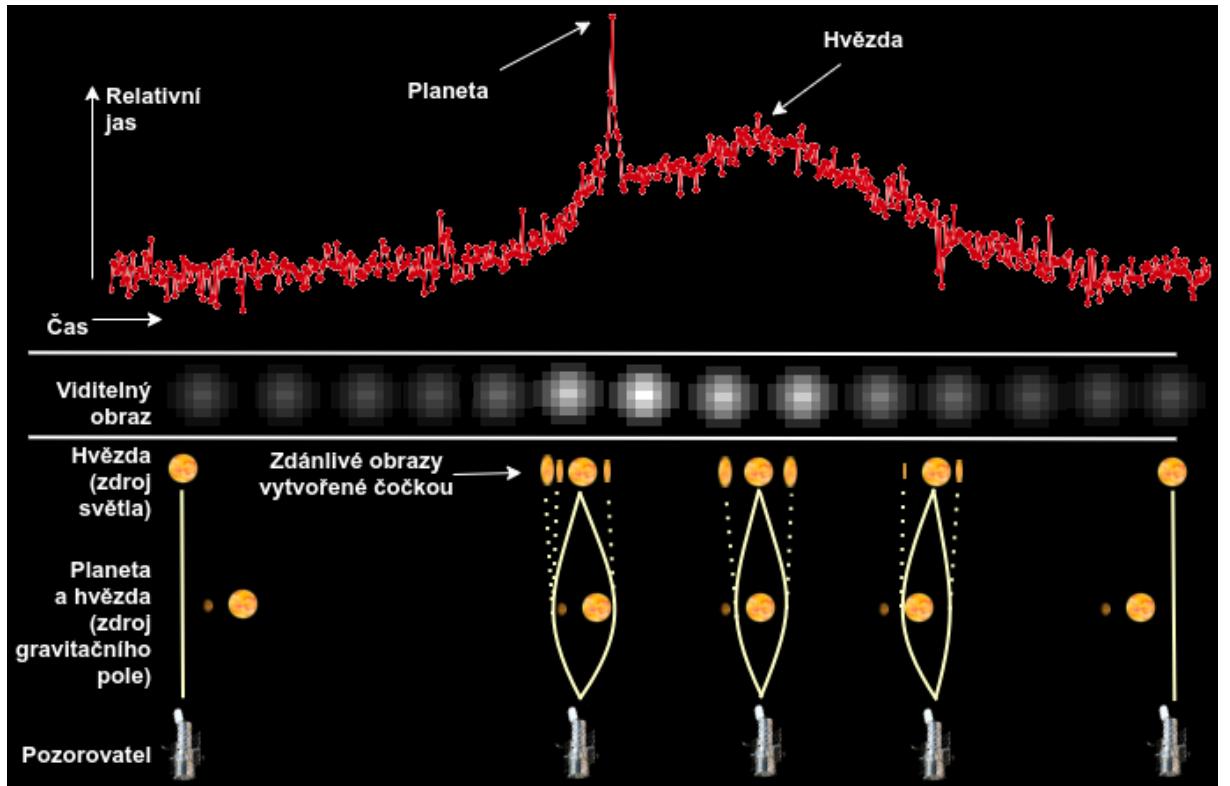


Obrázek 21: Princip gravitační čočky²

¹ Převzato z: [13].

² Vytvořeno autorem, fotografie gravitační čočky převzata od NASA (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), mluvíme o tzv. gravitační čočce – zdroj světla bude vidět vícekrát na různých místech, nebo bude naopak zdeformován. [13]



Obrázek 22: Ilustrace gravitační mikročočky OGLE 2003-BLG-235 ¹

Pokud je čočkujícím tělesem pouze hvězda, jedná se o gravitační mikročočku. V tomto případě dochází často ke splynutí 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, budeme pozorovat pozvolné zvyšování jasu způsobené hvězdou a zároveň v nějakém krátkém časovém intervalu strmý nárůst jasu způsobený exoplanetou. [13]

1.6 Časování pulsarů

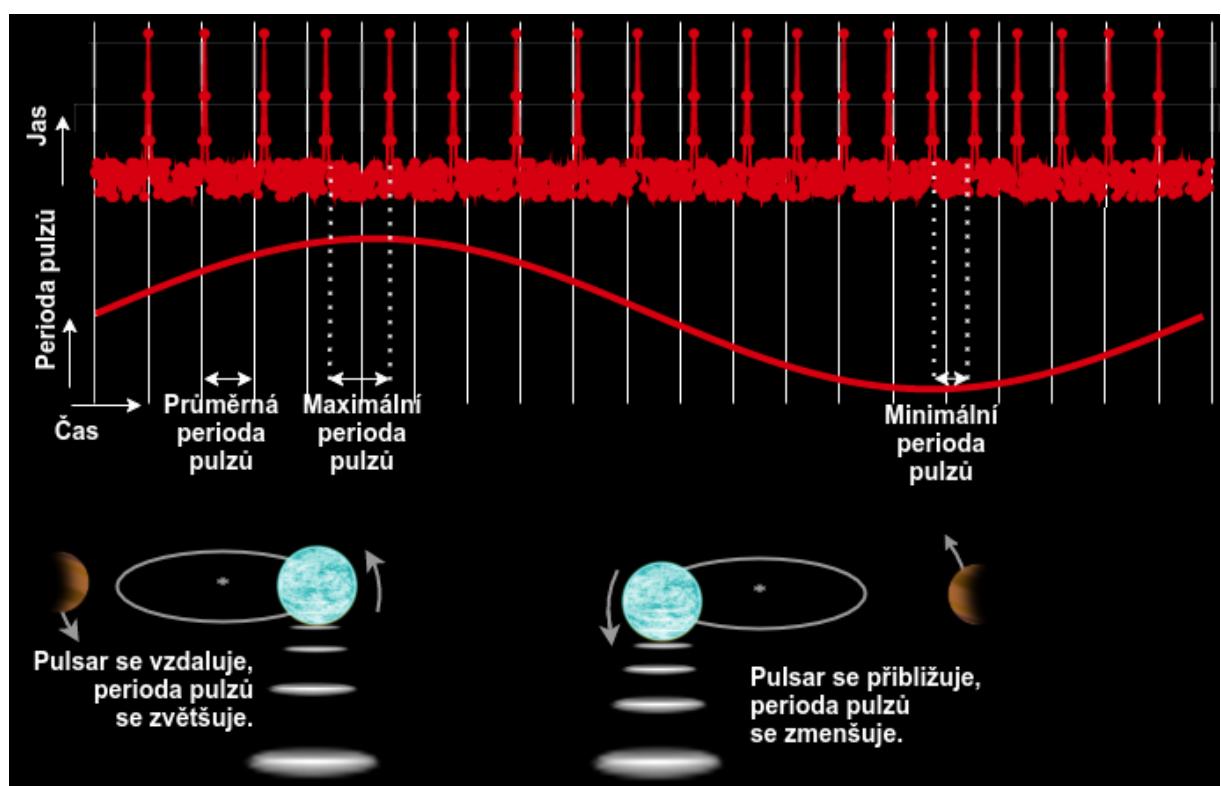
Pulsary jsou rychle rotující neutronové hvězdy – pozůstatky po zhrouceném jádře hmotných hvězd. Během procesu hrucení hvězda zmenšuje svůj poloměr a pro zachování momentu hybnosti zvyšuje svou rotační rychlosť. [13]

¹ Vytvořeno autorem, zdroj dat: [14].

Vedle toho, pulsar podél své magnetické osy emituje silný elektromagnetický paprsek. V případě, kdy je magnetická osa pulsaru natočena k Zemi (paprsek směruje k Zemi), můžeme zpozorovat náhlý nárůst jasu hvězdy. Tyto pulzy je pak možné sledovat ve velice pravidelných intervalech (často řádově jednotky milisekund až sekund). [13]

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. [13]

Nutno podotknout, že nás nezajímá rychlosť pulzaru tak, jako u metody radiálních rychlostí. Nepravidelnosti pulzů jsou způsobeny pouze polohou pulsaru, nikoliv jeho rychlosťí. Pulzy (elektromagnetické záření) se ve vakuu pohybují vždy konstantní rychlosťí bez ohledu na rychlosť pulzaru samotného (rychlosťi se nesčítají).



Obrázek 23: Ilustrace pulsaru s obíhající planetou ¹

¹ 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 jasu 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 můžeme chápat jako problém n těles, případně ji zjednodušit na problém 3 těles za předpokladu, že v soustavě jsou pouze 3 nezanedbatelně hmotná tělesa:

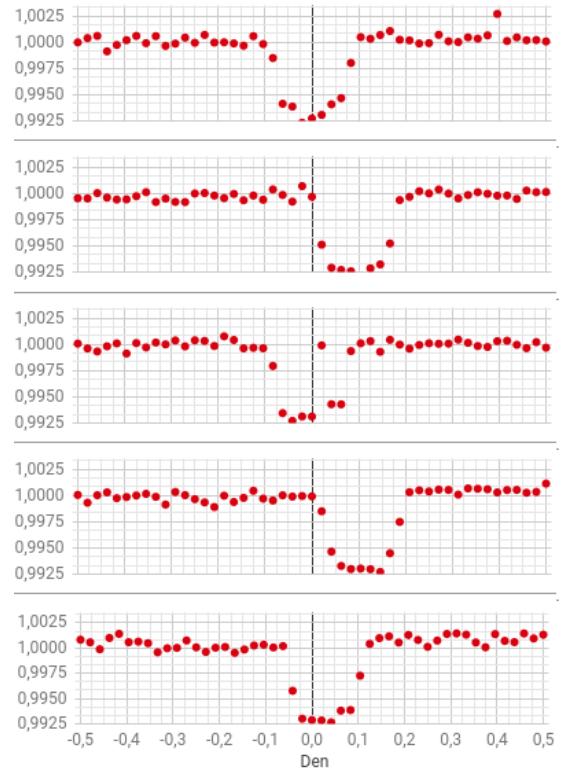
- Hvězda,
- Tranzitující planeta,
- Planeta způsobující nepravidelnosti.

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 anomálie) 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} , můžeme 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



Obrázek 24: Nepravidelnosti v tranzitech planety Kepler-46b¹

¹ Vytvořeno autorem, zdroj dat: [14].

funkce, tím přesnější řešení jsme našli. 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 popsány 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 rovnic znamých z problému 3 těles:

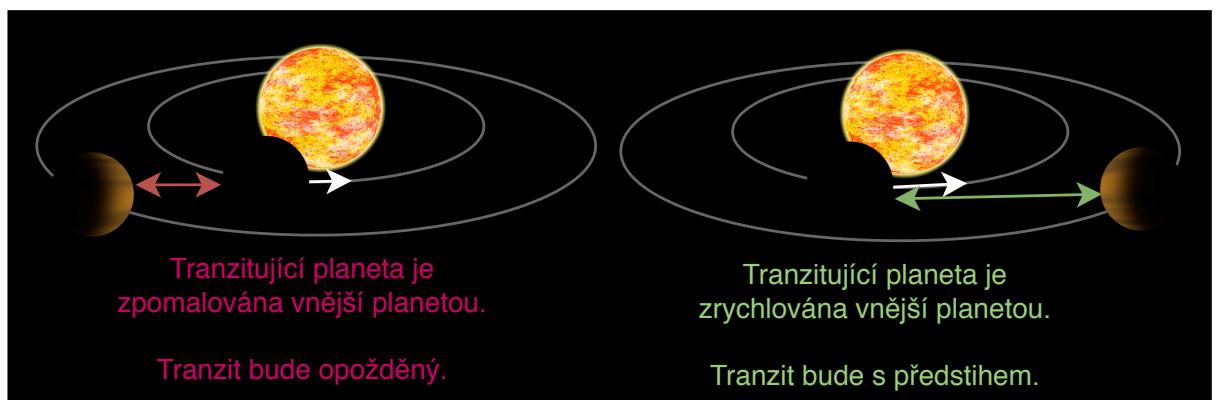
$$\ddot{r}_i = -Gm_j \frac{r_j - r_i}{|r_j - r_i|^3} - Gm_k \frac{r_k - r_i}{|r_k - r_i|^3} \quad (10)$$

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 $r_i = (x_i, y_i, z_i)$, jsme schopni 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 pokračujeme dále, abychom získali populaci potomků, kteří také 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 tranzitů¹

¹ 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

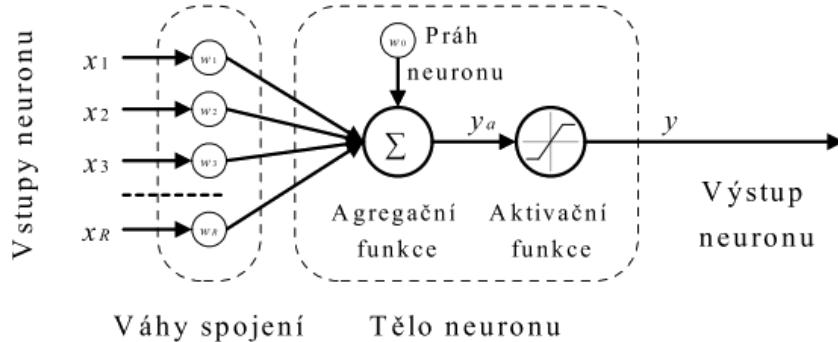
Základním stavebním prvkem umělé neuronové sítě je umělý neuron, který reprezentuje zjednodušenou biologickou nervovou buňku. Nejpoužívanějším typem umělého neuronu je tzv. formální neuron. Ten se skládá z následujících částí:

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

Tabulka 5: Komponenty formálního neuronu ¹

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

¹ Zdroj: [3]



Obrázek 26: Model formálního neuronu¹

2.1.1 Vstupy neuronu

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

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

Všechny vstupy neuronu můžeme 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]

¹ 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 Agregační funkce

Protože neuron může mít více vstupů, ale vždy pouze jediný výstup, je třeba všechny vstupy nějakým způsobem sloučit do jedné hodnoty. Právě toho se snaží docílit aggregační funkce neuronu. V případě formálního neuronu se nejčastěji používá prostá suma součinů vstupů s váhami. Výstupem aggregační funkce je potenciál 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: Agregační funkce neuronu

2.1.5 Aktivační funkce

Ještě než je potenciál y_a přiveden na výstup neuronu, je na něj aplikována aktivační funkce. Tato funkce je závislá na typu řešeného problému a stejně tak i na poloze neuronu v síti. Je běžné, že pro výstupní neurony se používají jiné aktivační funkce, než pro neurony ve skrytých vrstvách. Funkce mohou být lineární, nelineární, spojité i diskrétní. [3]

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

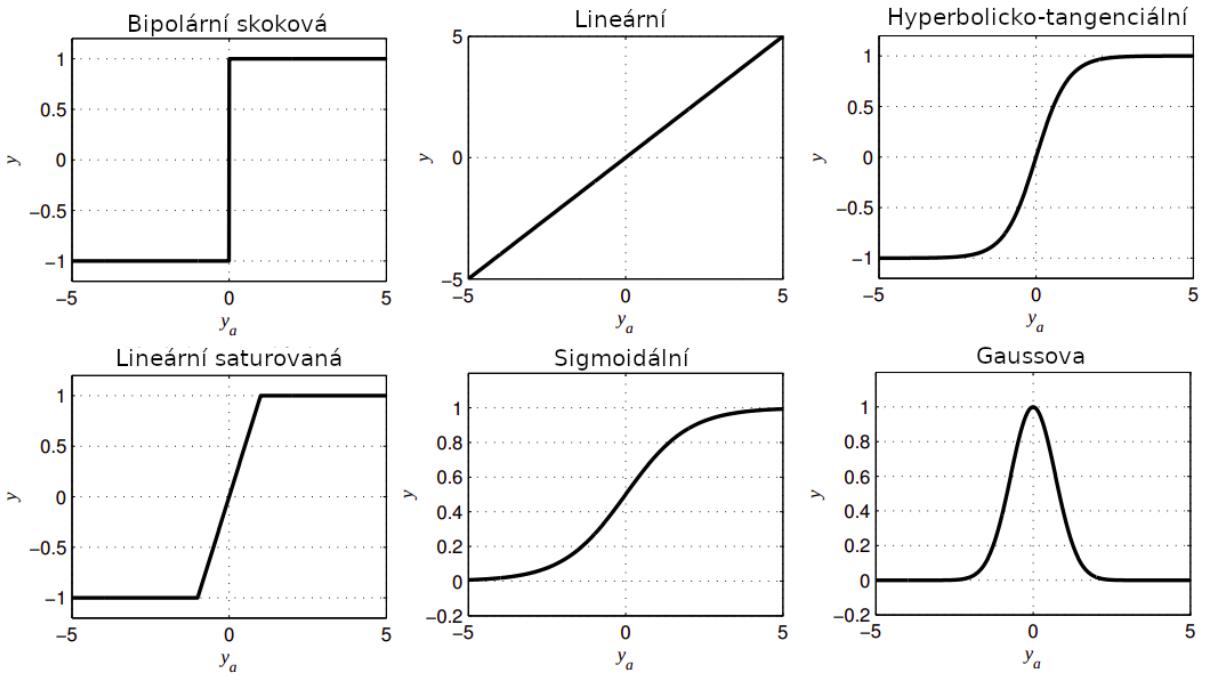
Vzorec 14: Hyperbolicko-tangenciální aktivační funkce

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

Vzorec 15: Sigmoidální aktivační funkce

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

Vzorec 16: Gaussova aktivační funkce



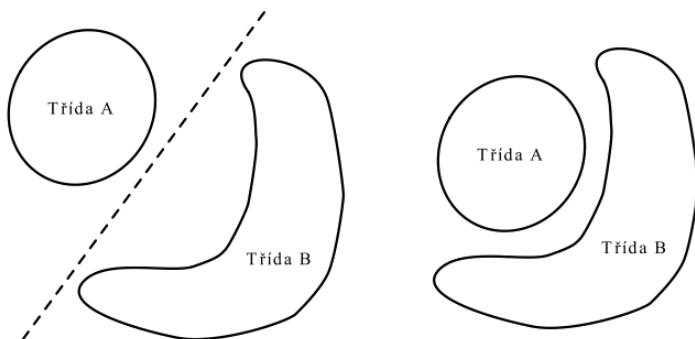
Obrázek 27: Příklady aktivačních funkcí neuronu ¹

2.2 Typy neuronových sítí

V této kapitole budou zmíněny některé typy neuronových sítí. Uvedený výčet však v žádém případě není kompletní. Vedle zmíněných typů existují např. rekurentní neuronové sítě nebo 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, učením s učitelem a skalárním výstupem nabývajícím binárních (1, 0) nebo bipolárních (1, -1) hodnot. [3]



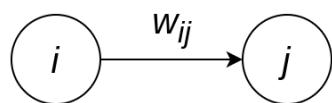
Obrázek 28: Lineárně separovatelná (vlevo) a neseparovatelná (vpravo) úloha ¹

¹ 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]

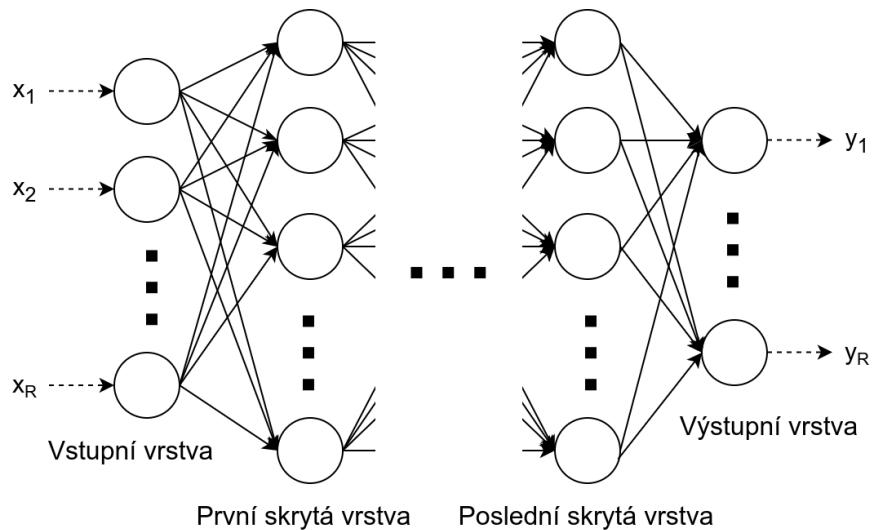
2.2.2 Dopředná vícevrstvá umělá neuronová síť

Dopředná vícevrstvá umělá neuronová síť (dále jen FFNN, někdy také vícevrstvý perceptron) je díky své univerzálnosti jedním z nejpoužívanějších typů sítě. 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 neuronů¹

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 sítě¹

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

¹ 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.

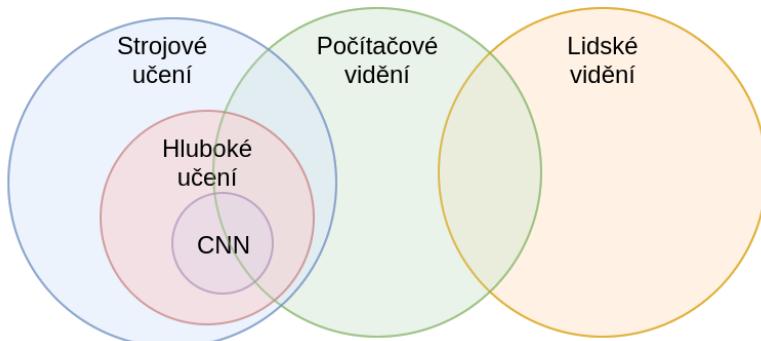


Obrázek 31: Ručně psané číslice 9.¹

Pro člověka je jednoduché rozpoznat, že všechny útvary na obrázku 25 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]



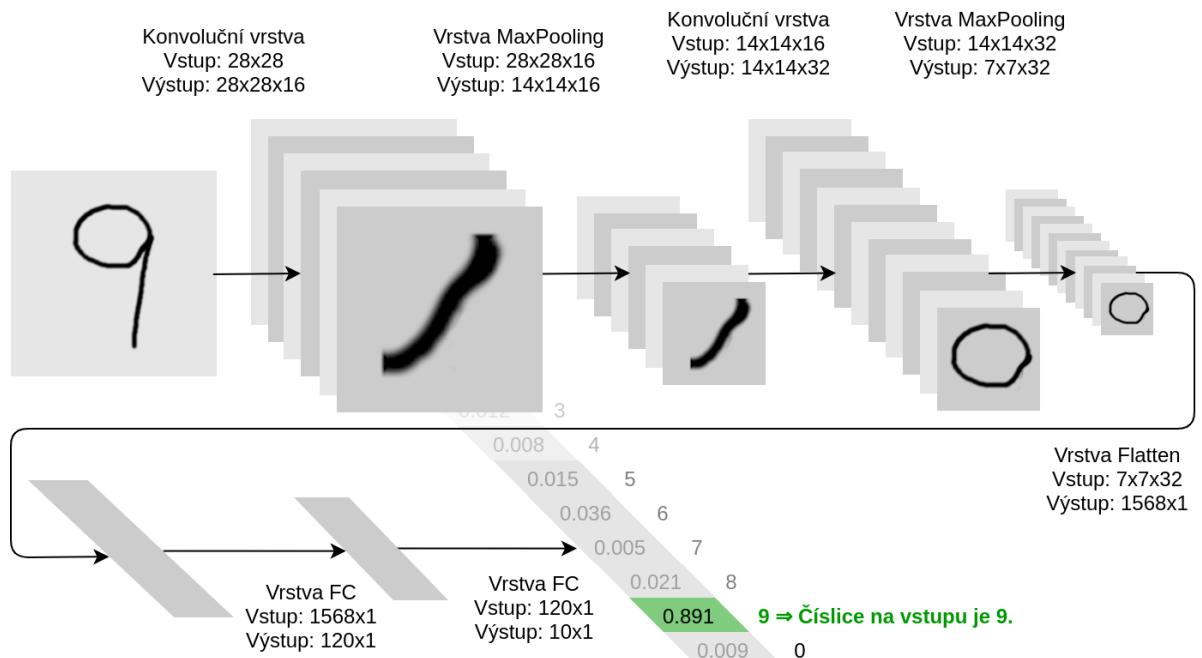
Obrázek 32: Vztah mezi počítačovým viděním, strojovým učením a konvoluční sítí²

¹ Převzato z: MNIST dataset.

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

My se zde budeme zabývat pouze konvoluční neuronovou sítí (CNN). Tu můžeme chápát 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 4.3.2.

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 4.3.3, která umožňuje zmenšit velikost vstupu a přitom zachovat vzory charakteristické pro vstup.



Obrázek 33: Příklad konvoluční sítě pro rozpoznávání ručně psaných číslic¹

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

¹ 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“.

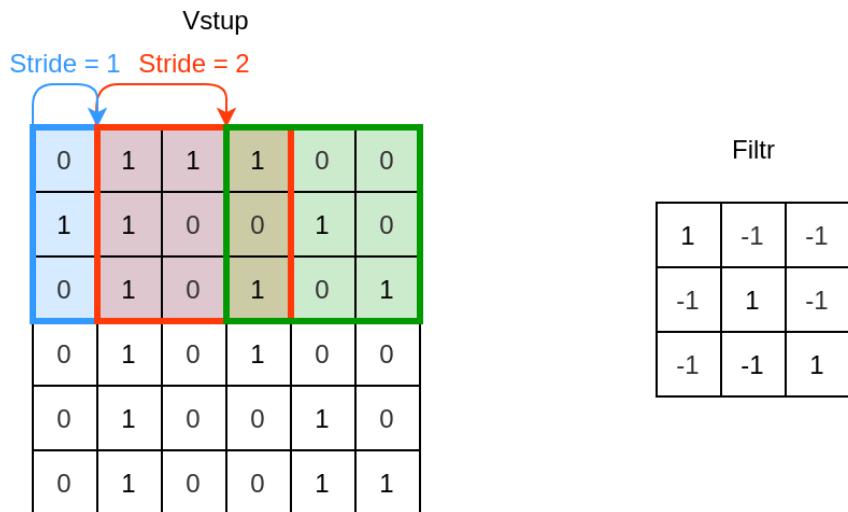
Parametr	Popis
Počet neuronů	Je určen složitostí a typem řešeného problému a určuje velikost výstupu vrstvy.

Tabulka 6: Parametry plně propojené vrstvy ¹

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“. Filtem můžeme chápout vzor, jenž je vyhledáván ve vstupu a který zastupuje nějakou vlastnost vstupního objektu (např. obrázku).



Obrázek 34: Postupné posouvání konvolučního filtru přes vstup ²

¹ Zdroj: [17]

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

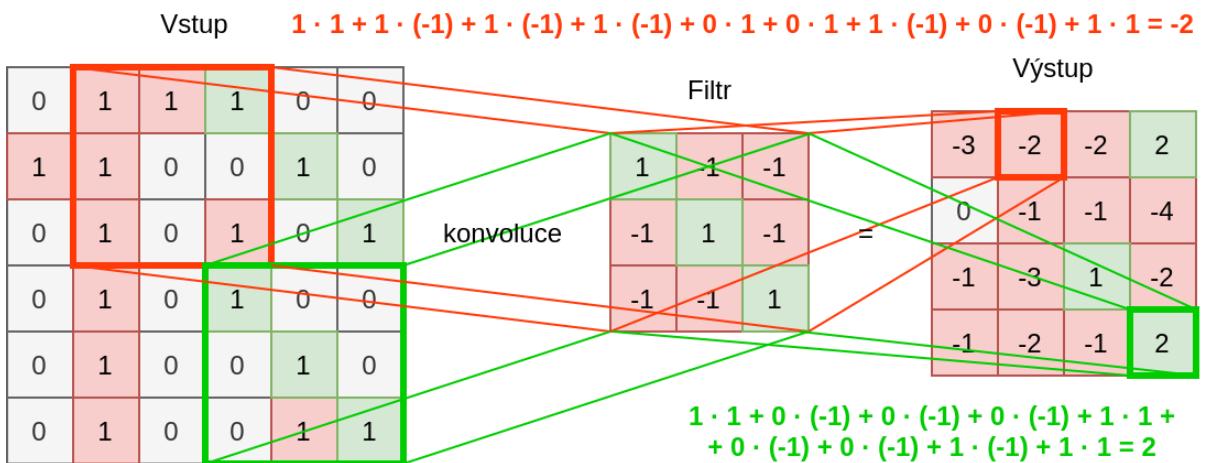
Pro výřez ze vstupu a filtr v konvoluční vrstvě se hodnoty na stejných pozicích ve výřezu i filtru vynásobí a tyto součiny se následně sečtou.

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

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 ¹

Na obrázku 29 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).
Posun filtrů (stride)	Posunutí filtru v každé fázi konvoluce.
Okraj (padding)	Způsob zpracování okrajů vstupu, které se ve výstupu buď useknou, nebo vyplní nějakou hodnotou.

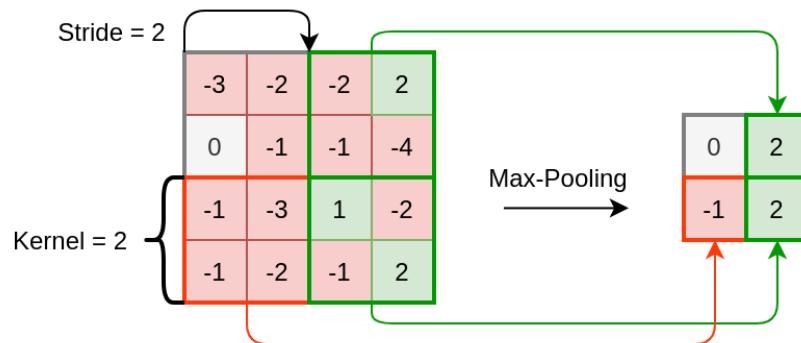
Tabulka 7: Parametry konvoluční vrstvy ²

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

² Zdroj: [17]

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, 17]



Obrázek 36: Příklad operace Max-Pooling¹

Na obrázku 30 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 máme 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-Pooling²

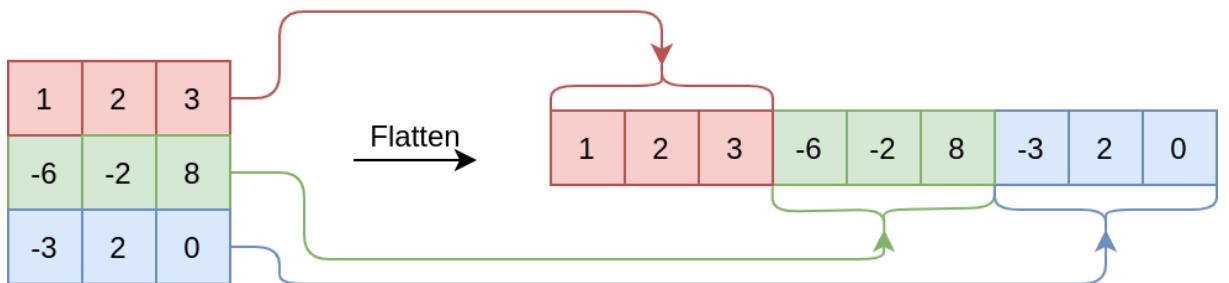
Obdobně k vrstvě Max-Pooling existuje např. i Min-Pooling nebo Average-Pooling. Jejich fungování je stejné s tím rozdílem, že na výstup předávají místo maximální hodnoty tu minimální, resp. průměrnou. Jejich použití je opět závislé na typu řešeného problému (např. zda vstupní obrázek ma světlé pozadí a tmavé popředí či naopak). [17]

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

² Zdroj: [17]

2.3.4 Vrstva Flatten

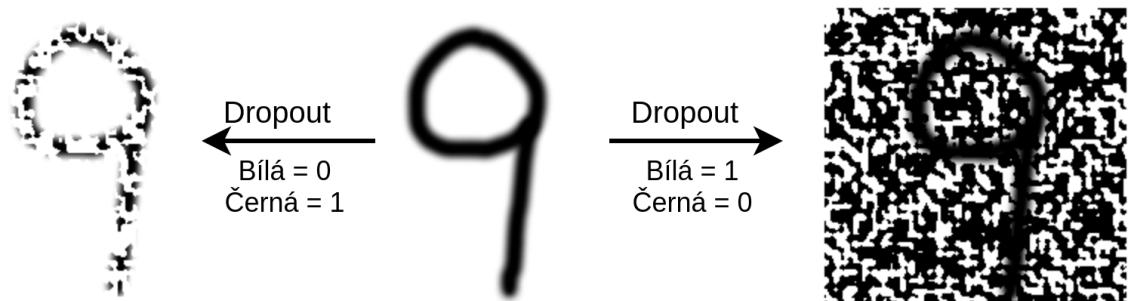
Konvoluční vrstvy umí pracovat s vícerozměrnými vstupy, vrstvy typu FC však očekávají pouze jednorozměrný vektor vstupních hodnot. Protože např. v konvolučních sítích je třeba oba typy vrstev kombinovat, je nutné nějakým způsobem převést n-rozměrný tenzor na vektor. Právě k tomu slouží vrstva Flatten. [17]



Obrázek 37: Příklad fungování vrstvy Flatten¹

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 vniká náhodný šum, což způsobí, že síť nebude trénovat pouze na stále stejných vstupech, ale trénovací množina bude vždy nepatrň odlišná – různorodější. [17]



Obrázek 38: Příklad fungování vrstvy Dropout¹

Parametr	Popis
Rate	Frekvence zahazování vstupů.

Tabulka 9: Parametry vrstvy Dropout²

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

² Zdroj: [17]

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 sítě je předložen vzor, pro který síť stanoví odezvu. Chyba pro jeden vzor je definována dle vzorce 15. [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

E = chyba jednoho vzoru	y_i = Skutečná hodnota i-tého výstupu	e_i = Chyba i-tého výstupu
Q = počet výstupů NN	t_i = Očekávaná hodnota i-tého výstupu	

Následně je ke každé váze připočten přírustek Δw_{ij}^k (při online učení – rychlejší) nebo se přírustek 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: Přírustek váhy u algoritmu zpětného šíření chyby

w_{ij}^k = Váha spoje mezi i-tým a j-tým neuronem v k-té vrstvě	α = rychlosť učenia
δ_j^k = lokální gradient neuronu aktualizované váhy	E = chyba jednoho vzoru
y_j^{k-1} = výstup j-tého neuronu v $k-1$. vrstvě	

Přičemž lokální gradient neuronu je vypočítán dle vzorce 17.

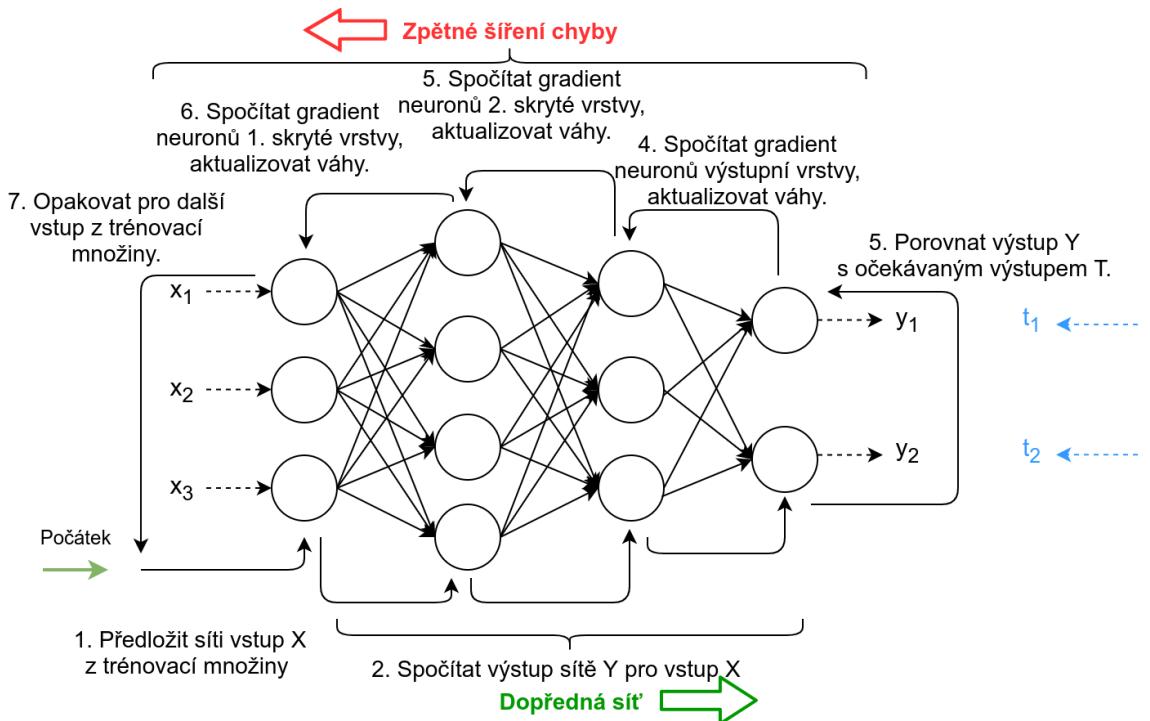
Tento postup je opakován pro všechny vzory v trénovací množině – této iteraci se říká *epocha*. Učení sestává z jedné nebo více epoch a může skončit:

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

Vzorec 20: Výpočet lokálního gradientu neuronu

δ_l^{k+1} = lokální gradient l-tého neuronu k + 1. vrstvy	e_j = chyba výstupu j-tého neuronu
y_{aj}^k = vstupní potenciál j-tého neuronu k-té vrstvy	ϕ^k = derivace aktivační funkce k-té vrstvy
w_{ij}^k = váha spoje mezi i-tým a j-tým neuronem k-té vrstvy	

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



Obrázek 39: Příklad algoritmu zpětného šíření chyby ¹

Počáteční hodnoty vah spojů mezi neurony mohou být nastaveny např. náhodně za použití rovnoměrného nebo normálního rozdělení s nulovou střední hodnotou. Rychlosť učení α určuje rychlosť, s jakou se vahy spojů mezi neurony upravují dle chyby při trénování. Obvykle nabývá hodnot jako 0,1, 0,01, 0,001, apod. [3]

Čím vyšší rychlosť 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]

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

3 DALŠÍ OBLASTI UMĚLÉ INTELIGENCE

Umělá inteligence (UI nebo AI) je podstatně rozsáhlý 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 inteligenčního chování strojů. [4] Lze ji rozdělit na 2 základní kategorie:

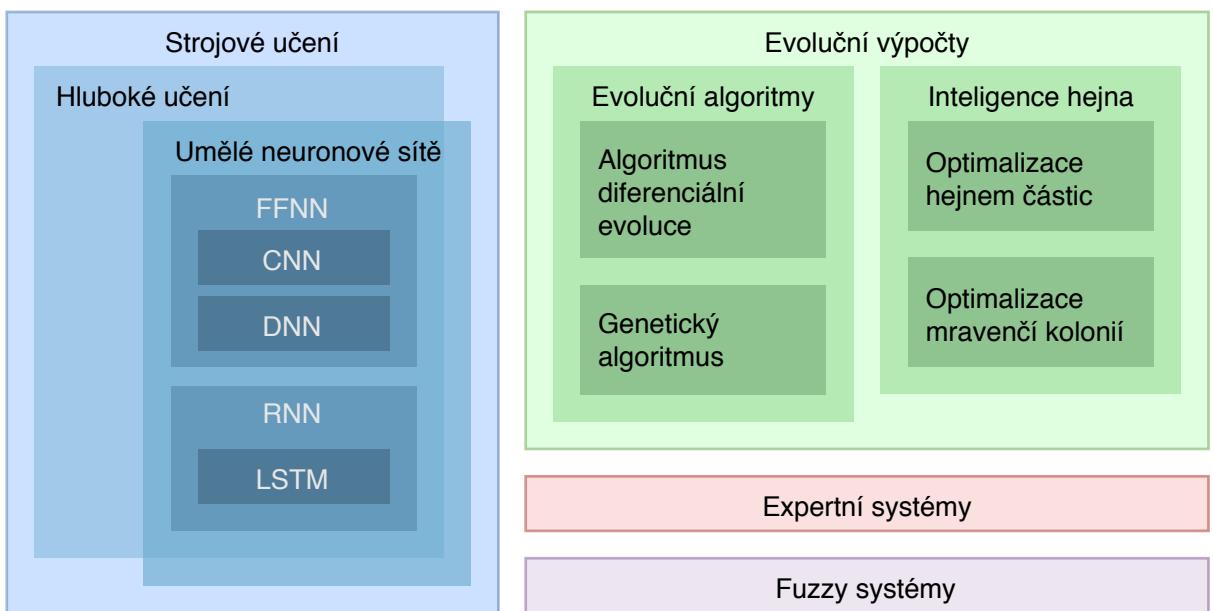
- **Silná umělá inteligence** – hypotetická obecná inteligence stroje umožňující řešit jakýkoli problém, který dokáže řešit člověk. Takový koncept však dosud reálně neexistuje, a proto zde nebude dále rozebíráno.
- **Slabá umělá inteligence** – skupina algoritmů, technik a postupů umožňujících řešit nějaký konkrétní a specifický problém neřešitelný nebo těžko řešitelný běžnými analytickými metodami, avšak často lehce řešitelný člověkem samotným.

Přičemž 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.
 - **Hluboké učení** – specifická část strojového učení spolehájící 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.
 - **Evoluční algoritmy** – snaží o napodobení evoluce a vychází z pravidla přežití nejsilnějšího. Často se zde vyskytují pojmy jako generace, přirozený výběr, dědičnost nebo křížení. Spadá sem např. genetický algoritmus nebo algoritmus diferenciální evoluce.
 - **Inteligence 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 nenumerické 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 té lidské.



Obrázek 40: Rozdělení umělé inteligence¹

3.1 Genetický algoritmus

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

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 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.

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

```

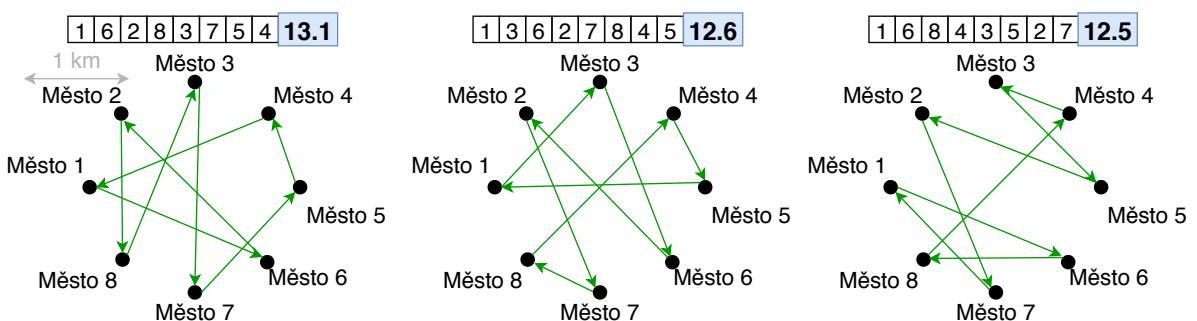
1. Define K (population size), I (number of generations), M (mutation probability) a F (fitness function).
2. Generate K random individuals (first generation).
3. Apply F to all individuals in first generation.
4. Repeat from 1 to I:
    4.1. Repeat from 1 to K:
        4.1.1. Select two random parents  $x_1$ ,  $x_2$  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). V případě TSP pro 8 měst může být jedinec reprezentován např. jako sekvence měst, která budou po stupně navštívěna. 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: Některí z vygenerovaných jedinců GA pro TSP ¹

Na počátku běhu algoritmu dojde k náhodnému vygenerování K jedinců – ti budou tvořit populaci první generace. Počet jedinců K v populaci volíme na základě složitosti řešeného problému. Čím vyšší je K, tím déle bude zpravidla běh algoritmu trvat, avšak tím slabilnější bude a tím spíše najde lepší řešení.

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

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.

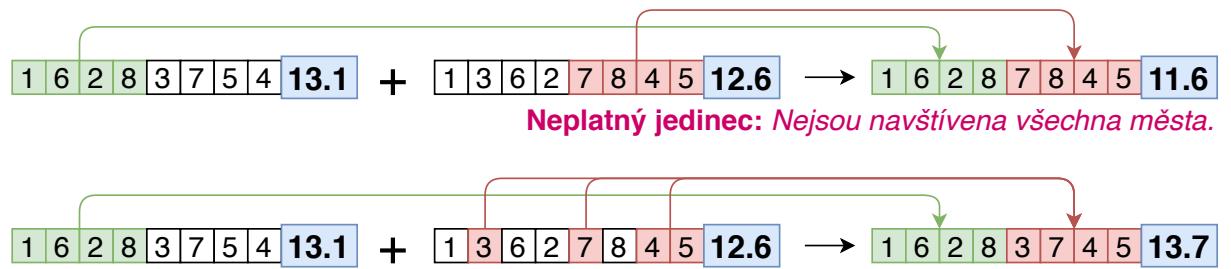
3.1.2 Selekce

Přirozený výběr je založen na skutečnosti, že úspěšnější (silnější) jedinci mají vyšší pravděpodobnost předat své geny a rozmnožit se. Abychom tento koncept mohli napodobit i v genetickém algoritmu, potřebujeme definovat funkci F , která dokáže posoudit kvalitu jedince – tzv. fitness.

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 jsme vybrali jedince 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.



Obrázek 42: Příklad křížení jedinců GA pro TSP ¹

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

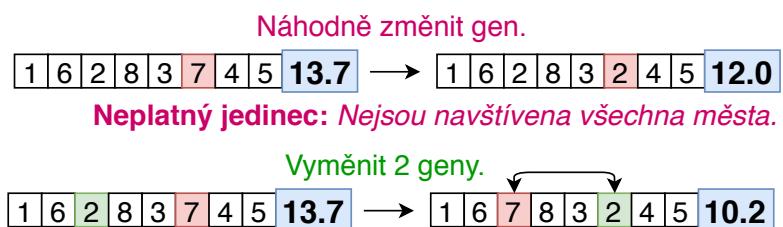
¹ Vytvořeno autorem v <https://www.draw.io>.

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.

3.1.4 Mutace

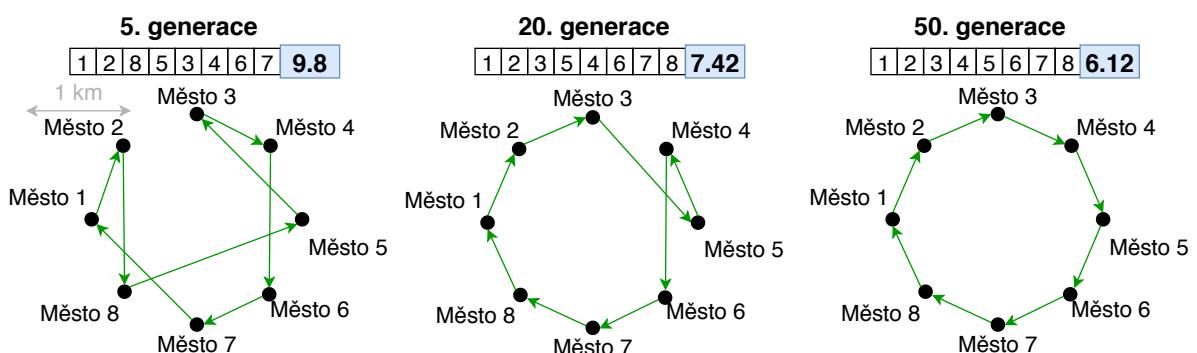
Nyní, když máme další generaci jedinců, je vhodné zajistit vyšší rozmanitost populace. Toho lze docílit tak, že každý jedinec má určitou pravděpodobnost, že zmutuje – dojde k náhodné modifikaci jeho genů.



Obrázek 43: Příklad mutace jedince GA pro TSP¹

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

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.



Obrázek 44: Hledání řešení TSP pomocí GA¹

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

4 PROJEKTY PRO HLEDÁNÍ EXOPLANET

4.1 Planet Hunters

TODO

4.2 Astronet

TODO: [11]

5 POUŽITÉ TECHNOLOGIE

5.1 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.

5.1.1 React

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

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

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

Zdrojový kód 2: Ukázka práce s knihovnou React

5.1.2 Styled components

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

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

Zdrojový kód 3: Ukázka práce s knihovnou Styled components

5.2 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 Java). 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.

5.2.1 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 tzv. extensions.

```
from flask import Flask

app = Flask(__name__)

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

Zdrojový kód 4: Ukázka práce s microframeworkem Flask

5.2.2 Astropy

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

```
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

5.2.3 LightKurve

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

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

tpf = tpf.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

5.2.4 Keras

Keras (<https://keras.io/api>) je vysokoúrovňová nádstavba nad TensorFlow pro práci s umělými neuronovými sítěmi. Poskytuje jednoduché API a možnost rozšířitelnosti.

```
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

5.2.5 MongoEngine

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

```

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

5.3 MongoDB

MongoDB (<https://docs.mongodb.com>) je multiplatformní dokumentová NoSQL databáze. Ve verzi zdarma přes clouдовou 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.

```

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

5.4 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>).

```

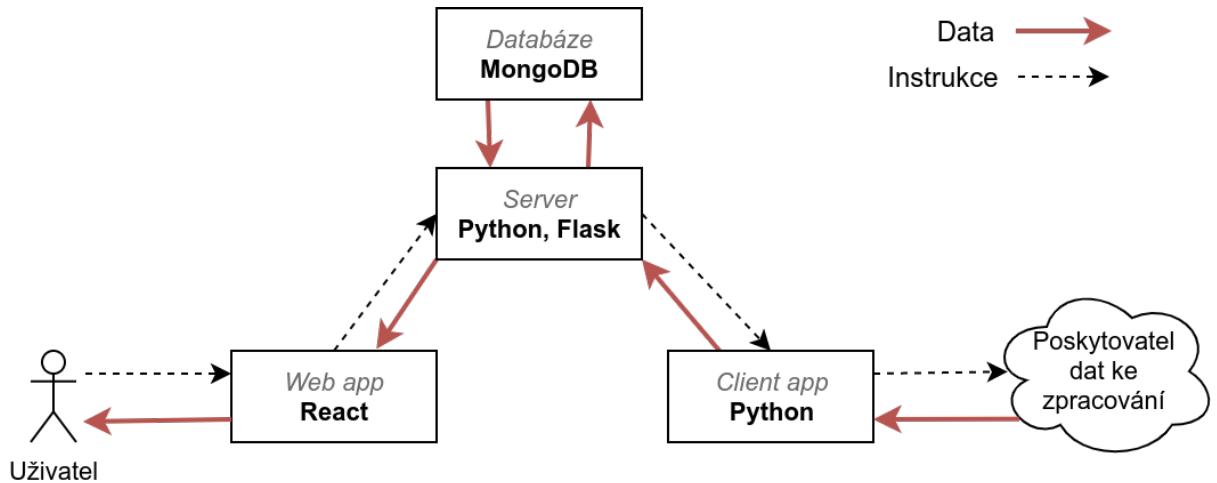
# Client:
@socketio.event
def process(task):
    print(f"Data processing from address {task['url']}...")

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

```

Zdrojový kód 10: Ukázka komunikace pomocí socket.io.

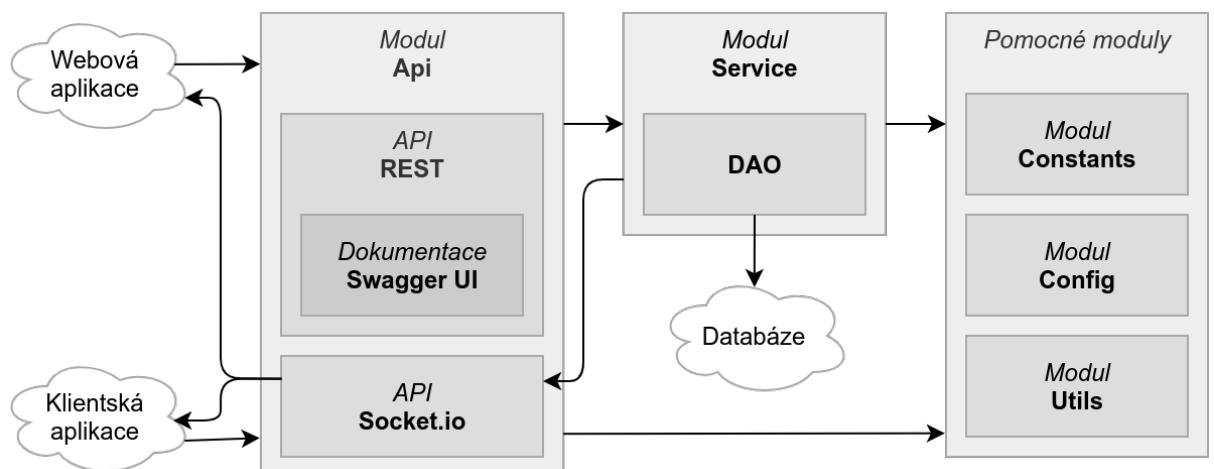
6 NÁVRH A VÝVOJ APLIKACE



Obrázek 45: Komponenty projektu a jejich komunikace ¹

6.1 Server

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



Obrázek 46: Architektura serveru ¹

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

¹ 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 (`./src/main.py --env test`). [TODO]
- **constants** – Veškeré konstantní položky, ať už technického či fyzikálního typu.
- **service** – Servisní vrstva obsahující logiku serveru a manipulaci s databází.
- **utils** – Pomocné třídy uchovávající specifickou funkcionalitu.

6.1.1 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 také pro ně nastavit 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.

```
@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)

```

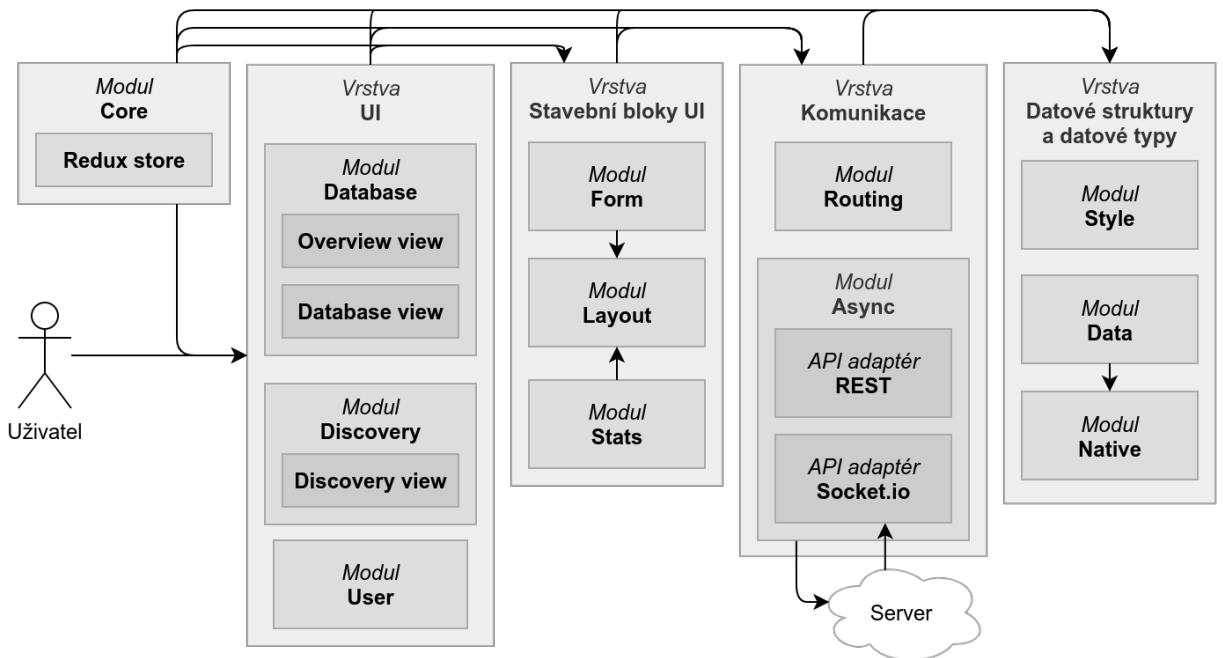
Zdrojový kód 12: Vytvoření koncového bodu v REST API.

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 [TODO] na konci dokumentu.

6.1.2 Socket.io API

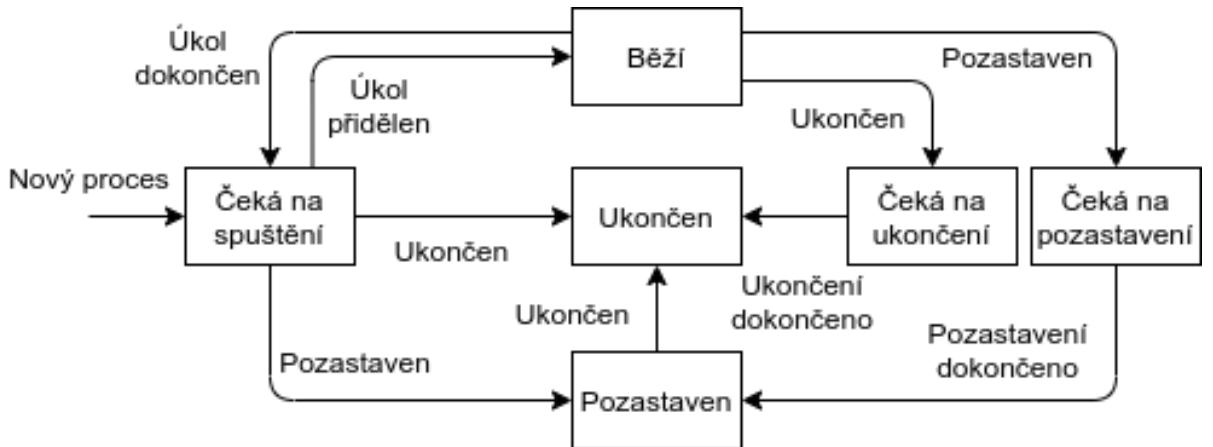
Protože je třeba v reálném čase synchronizovat webovou a klientskou aplikaci a umožnit serveru iniciovat s těmito aplikacemi komunikaci, server kromě rozhraní HTTP poskytuje také rozhraní Socket.io.

6.2 Webová aplikace



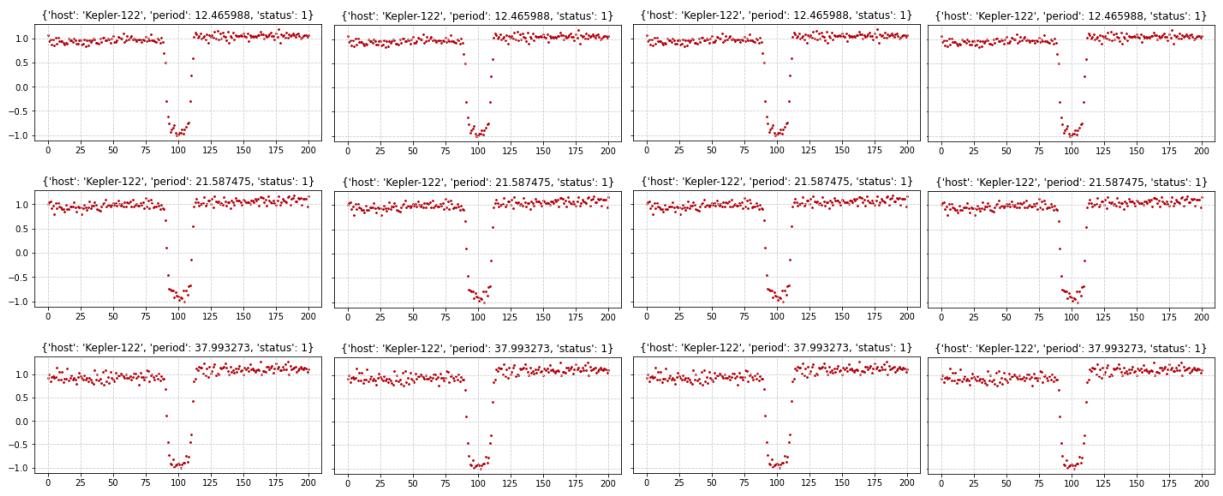
Obrázek 47: Architektura webové aplikace ¹

6.3 Klientská aplikace



Obrázek 48: Stavy procesů a přechody mezi nimi¹

6.4 Neuronová síť



Obrázek 49: Trénovací množina¹

TODO: Odebrat osy a popisky z obrázku.

6.5 Databáze

6.6 Datasety

Většina dat se do databáze nevkládá manuálně, jelikož je třeba pracovat se stovkami tisíc či miliony datových položek. Namísto toho administrátor pouze definuje přístupový bod

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

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

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 dál zpracovávat. Zpracování jiných je naopak výpočetně náročné a připojené klientské aplikace tak činí postupně položku po položce. Jednotlivé typy datasetů jsou popsány níže.

U všech typů datasetů je také vyřešena situace, kdy by se jedna a ta samá položka (např. hvězda) nacházela ve dvou datasetech zároveň. V takovém případě dojde k uložení obou hodnot a uživatel v aplikaci pak bude vidět údaje dané položky z obou datasetů vedle sebe.

6.6.1 Target pixel files

6.6.2 Světelné křivky hvězd

6.6.3 Radiální rychlosti hvězd

6.6.4 Hvězdy

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



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 50: Část datasetu s vlastnostmi hvězd ¹

Tyto datasety 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):

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

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. Datasety tak mohou mít pořadí i názvy jednotlivých sloupců libovolné.

6.6.5 Planety

Datasety s planetami nejsou pro běh aplikace taktéž nutné, protože veškeré informace o exoplanátech 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 datasety s údaji o planetách – mohou potvrdit nebo vyvrátit údaje vypočtené v rámci aplikace.

pl_hostname	pl_letter	pl_controvflag	pl_pnum	pl_orbper	pl_orbperlim	pl_orbperrn	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 51: Část datasetu s vlastnostmi planet¹

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	Rychlosť oběhu	Podmínky pro život

Tabulka 11: Údaje o planetách ukládané do databáze

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

6.6.6 Názvy

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

Katalog	Účel	Označení Kepler-10
KIC (K epler I nput Catalog)	Hledání exoplanet (Kepler)	KIC 11904151
KOI (K epler O bject of I nterest)	Výběr z KIC	KOI-72
Kepler	Potvrzené exoplanety z KOI	Kepler-10
2MASS (2 Micron A ll- S ky S urvey)	Infra. průzkum oblohy	2MASS J19024305+5014286
GSC (G uide S tar C atalog)	Pozorování hvězd (Hubble)	GSC 03549-00354
Gaia DR (G aia D ata R 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 ²

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.

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

7 ROZVRŽENÍ APLIKACE

7.1 Přehled

7.2 Databáze

7.3 Detail systému

Na jediné stránce jsou shrnutý všechny známé informace o systému. Kromě hodnot veličin hvězdy a případných planet jsou zde zaznamenána všechna pozorování, vizuální porovnání velikostí a vzdáleností oproti sluneční soustavě, seznam referencí a v neposlední řadě také interaktivní 3D model systému.

7.4 Objevování

7.5 Nápověda

7.6 Autentizace

ZÁVĚR

TODO: Verifikace výsledků.

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] VRAHATIS, Michael. Computational Approaches to Artificial Intelligence: Theory, Methods and Applications.. *Springer*. [online]. 2019. [cit. 28. 11. 2020]. Dostupné z: https://www.researchgate.net/publication/334023205_Computational_Approaches_to_Artificial_Intelligence_Theory_Methods_and_Applications
- [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
- [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://citeseervx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.4155&rep=rep1&type=pdf>

- [9] OGBUEFI, Kelvin. 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>
- [10] 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>
- [11] 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>
- [12] 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>
- [13] Metody objevování planet. *Astronomia*. [online]. 23. 1. 2013. [cit. 23. 12. 2019]. Dostupné z: <http://hvezdy.astro.cz/exoplanety/51-metody-objevovani-planet>
- [14] NASA Exoplanet Archive. *NASA Exoplanet Science Institute*. [online]. 12. 8. 2019. [cit. 25. 12. 2019]. Dostupné z: https://exoplanetarchive.ipac.caltech.edu/docs/API_exoplanet_columns.html
- [15] Reference Solar Spectral Irradiance: ASTM G-173. *nrel*. [online]. ???. ?. ?????. [cit. 27. 12. 2019]. Dostupné z: <https://rredc.nrel.gov/solar/spectra/am1.5/ASTMG173/ASTMG173.html>
- [16] TESS Exoplanet Mission. *NASA*. [online]. 24. 8. 2020. [cit. 8. 10. 2020]. Dostupné z: <https://www.nasa.gov/content/about-tess>
- [17] Keras API reference. *Keras*. [online]. 2020. [cit. 29. 10. 2020]. Dostupné z: <https://keras.io/api>

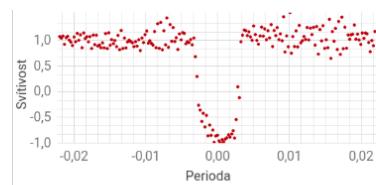
Exoplanets data http://exoplanets.org/detail/alpha_Cen_B_b
 Wavelength <http://spiff.rit.edu/classes/phys240/lectures/expand/expand.html>
 TODO: Doplnit reference ze zadání diplomové práce.

SEZNAM PŘÍLOH

Příloha A Architektura neuronové sítě	69
---	----

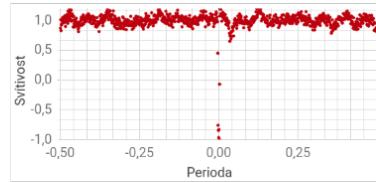
PŘÍLOHA A – ARCHITEKTURA NEURON. SÍTĚ

Conv1D		<i>Input: 1001x1</i>
Filters: 16	Kernel: 3	<i>Output: 1001x1</i>
Stride: 1	Activation: ReLu	



Conv1D		<i>Input: 1001x1</i>
Filters: 16	Kernel: 3	<i>Output: 1001x1</i>
Stride: 1	Activation: ReLu	

MaxPool1D		<i>Input: 1001x1</i>
Pool: 2	Stride: 1	<i>Output: 999x16</i>



MaxPool1D		<i>Input: 1001x1</i>
Pool: 2	Stride: 1	<i>Output: 999x16</i>

Conv1D		<i>Input: 1001x1</i>
Filters: 16	Kernel: 3	<i>Output: 1001x1</i>
Stride: 1	Activation: ReLu	

Conv1D		<i>Input: 1001x1</i>
Filters: 16	Kernel: 3	<i>Output: 1001x1</i>
Stride: 1	Activation: ReLu	

MaxPool1D		<i>Input: 1001x1</i>
Pool: 2	Stride: 1	<i>Output: 999x16</i>

MaxPool1D		<i>Input: 1001x1</i>
Pool: 2	Stride: 1	<i>Output: 999x16</i>

Conv1D		<i>Input: 1001x1</i>
Filters: 16	Kernel: 3	<i>Output: 1001x1</i>
Stride: 1	Activation: ReLu	

Conv1D		<i>Input: 1001x1</i>
Filters: 16	Kernel: 3	<i>Output: 1001x1</i>
Stride: 1	Activation: ReLu	

MaxPool1D		<i>Input: 1001x1</i>
Pool: 2	Stride: 1	<i>Output: 999x16</i>

MaxPool1D		<i>Input: 1001x1</i>
Pool: 2	Stride: 1	<i>Output: 999x16</i>

Flatten		<i>Input: 1001x1</i>
		<i>Output: 1001</i>

Flatten		<i>Input: 1001x1</i>
		<i>Output: 1001</i>

Concatenate		
<i>Input: 1001x1</i>		<i>Output: 1001</i>

Dense		<i>Input: 1001x1</i>
Neurons: 16	Activ.: Tanh	<i>Output: 999x16</i>

Dropout		<i>Input: 1001x1</i>
		<i>Output: 1001</i>

Dense		<i>Input: 1001x1</i>
Neurons: 16	Activ.: Tanh	<i>Output: 999x16</i>

PŘÍLOHA B – ARCHITEKTURA DATABÁZE

