České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra elektroenergetiky

Analýza nasazení umělé inteligence na využití přebytků v domácím fotovoltaickém systému

Diplomová práce

Bc. Josef Kubička

Studijní program: Elektrotechnika, energetika a management
Studijní obor: Elektroenergetika
Vedoucí práce: Doc. Mgr. Jakub Holovský, Ph.D.

Květen 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kubička** Jméno: **Josef** Osobní číslo: **483505**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektroenergetiky**

Studijní program: **Elektrotechnika, energetika a management**

Specializace: **Elektroenergetika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Analýza nasazení umělé inteligence na využití přebytků v domácím fotovoltaickém systému

Název diplomové práce anglicky:

Analysis of the artificial intelligence implementation in the use of excess solar energy in household system

Pokyny pro vypracování:

- Proveďte rešerši využití umělé inteligence pro řízení mikrosítí.
- Definujte pravidla využití přebytků v domácnosti.
- Implementujte vybranou metodu umělé inteligence.
- Na naměřených datech proveďte trénování algoritmu.
- Proveďte test řízení využití přebytků v domácím FV systému.
- Simulujte řízení domácího fotovoltaického systému pomocí komerčního softwaru.
- Srovnejte obě předcházející řešení.

Seznam doporučené literatury:

[1] S. Sen and V. Kumar, “Microgrid control: A comprehensive survey,” Annual Reviews in Control, vol. 45, pp. 118–151, 2018, doi: 10.1016/j.arcontrol.2018.04.012.
[2] R. Trivedi and S. Khadem, “Implementation of artificial intelligence techniques in microgrid control environment: Current progress and future scopes,” Energy and AI, vol. 8, p. 100147, May 2022, doi: 10.1016/j.egyai.2022.100147.
[3] A. Sahithi, „Simulace FV systému s ukládáním energie do vody“. České vysoké učení technické v Praze, 01/2022. https://dspace.cvut.cz/handle/10467/99200
[4] M. Černohorský, „Využití přebytků energie domácího fotovoltaického systému pomocí zařízení IoT“. České vysoké učení technické v Praze, 05/2022. https://dspace.cvut.cz/handle/10467/101318

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Mgr. Jakub Holovský, Ph.D. katedra elektrotechnologie FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **17.02.2023** Termín odevzdání diplomové práce: **26.05.2023**

Platnost zadání diplomové práce: **22.09.2024**

doc. Mgr. Jakub Holovský, Ph.D.	doc. Ing. Zdeněk Müller, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 26. května 2023

.....
Bc. Josef Kubička

Abstrakt

Diplomová práce se zbývá problematikou aplikace umělé inteligence v energetice. V úvodu jsou popsány mikrosítě a virtuální elektrárny, které mohou být v budoucnosti velmi důležitými prvky elektrizační soustavy. Součástí popisu je analýza možnosti jejich řízení včetně implementace nástrojů umělé inteligence. Podrobněji jsou rozebrány umělé neuronové sítě a způsoby jejich učení. V následující části jsou praktické implementace umělých neuronových sítí. Nejprve je ukázka návrhu mělké neuronové sítě pro optimalizaci domácí fotovoltaické elektrárny. Poté následuje návrh systému pro řízení ohřevu vody pomocí zpětnovazebního učení. V závěru je vyhodnocen potenciál implementovaných metod.

Klíčová slova: Umělá inteligence, Mikrosítě, Virtuální elektrárny, Umělé neuronové sítě, Zpětnovazební učení

Abstract

Diploma thesis studies the aplication of artificial inteligence in power engineering. It starts with a description of microgrids and virtual power plants, which have a potential to be very important part of the future electric power systems. The description includes their control systems and the means of implementing artificial inteligence. Artificial neural networks and their learning processes are studied in more detail. The following part consists of practical implementation of artificial neural networks. Firstly, there is a an example of designing a shallow neural network for solar system optimization. Second implementation is creating a system for water heating control using reinforcement learning. The thesis ends with an assesment of used methods potential.

Keywords: Artificial inteligence, Microgid, Virtual power plant, Artificial neural network, Reinforcement learning

Poděkování

Tímto bych chtěl poděkovat vedoucímu této práce panu docentovi Jakubovi Holovskému za odborné vedení, aktivní přístup a mnoho cenných nápadů a rad.

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Umělá inteligence je v současnosti velmi populárním tématem. Díky své univerzálnosti nalezne využití v každém technickém odvětví a energetika není výjímkou. Cílem práce je ukázat, kam by mohl směřovat budoucí vývoj v energetice a perspektivních technologií, které tento rozvoj podpoří. Proto se v úvodní části práce budu zabývat mikrosítěmí a virtuálními elektrárnami a metodami jejich řízení, u kterých lze tyto technologie využít. Jakožto praktickou ukázku se budu věnovat aplikaci umělé inteligence při návrhu, analýze a řízení fotovoltaického systému. Nejprve ukáži, jak postupovat při návrhu umělé neuronové sítě a jak ji využít například pro optimalizaci fotovoltaické elektrárny. Poté se budu zabývat návrhem řídícího systému ohřevu vody přebytky z fotovoltaiky pomocí zpětnovazebního učení. Všechny praktické implementace jsou koncipovány tak, aby je mohl kdokoliv zopakovat bez nutnosti vysoce výkonných výpočetních jednotek. Fotovoltaiku jsem vybral proto, jelikože bude v nedaleké budoucnosti pravděpodbně nejrozšířenějším zdrojem elektrické energie.

V oblasti umělé inteligence se používá mnoho anglických termínů, které se často nepřekládají, jelikož je jejich překlad nevýstižný a vzhledem k podstatně většímu množství materiálů zabývajících se touto problematikou v angličtině považuji znalost anglických termínů vhodnější než znalost termínů českých. Proto v této práci uvádím anglické ekvivalenty včetně jejich typických zkratek a v některých případech zmíním pouze termín anglický.

Kapitola 1

Mikrosítě

Mikrosíť (anglicky Microgrid, zkráceně μgrid nebo MG) je organizovaný energetický systém složený z několika decentralizovaných zdrojů elektrické energie (Distributed energy resources, DERs), a to obnovitelných i konvenčních. Oproti konvenčnímu způsobu generace elektřiny ve velkých elektrárnách a následnému přenosu přes elektrická vedení dlouhá mnoho kilometrů nabízí mikrosítě mnoho výhod, mezi které patří vyšší lokální spolehlivost dodávky elektrické energie, vyšší kvalita energie, nižší emise díky snazší integraci obnovitelných zdrojů nebo vyšší účinnost díky menším přenosovým ztrátám na podstatně kratších vedeních. Nevýhodou je potřeba komplexních rozhraní výkonové elektroniky (Power electronic interface, PEI) a řídících systémů pro každý decentralizovaný zdroj, které musí zajistit, že se celý systém decentralizovaných zdrojů energie chová obdobně jako konvenční zdroje z hlediska spolehlivosti, stability a kvality elektrické energie.

Vzhledem ke komplexním požadavkům na řízení je aplikace prvků inteligentních systémů pro efektivní provoz téměř nutností. Zároveň se jedná o perfektní příležitost pro testování těchto technologií v menším měřítku před přechodem na celou distribuční nebo přenosovou síť. Přestože jsou mikrosítě rozsáhle studovány a testovány, jejich komerční využití zatím naráží na řadu překážek z hlediska technologického, ekonomického i regulatorního.

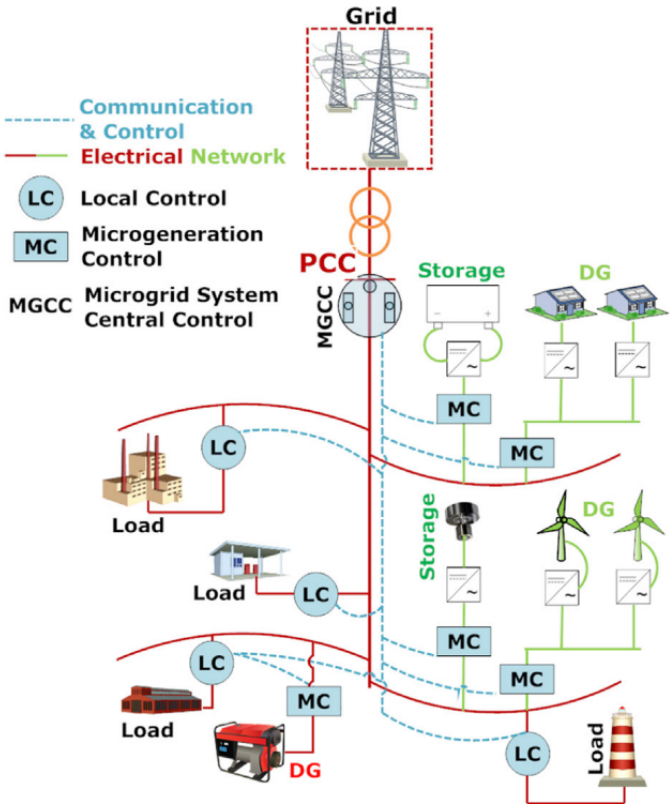
1.1 Architektura mikrosítí

Mikrosíť se skládá ze čtyř základních částí:

- distribuční systém,
- decentralizované zdroje,
- úložiště energie,
- řídící a komunikační systémy.

Graficky je obecná architektura mikrosítě znázorněna na obr. 1.1. Tento obrázek ukazuje typický příklad mikrosítě, mikrosítí ale existuje mnoho druhů právě z hlediska své struktury. Mikrosíť může být připojená k distribuční sítí, anebo být zcela samostatná a pracovat v ostrovním režimu. Z hlediska typu napětí se standardně dělí na stejnosměrné (DC) a střídavé (AC). Dále můžeme mikrosítě roztřídit podle typu decentralizovaných zdrojů (obnovitelné, konvenční atd.) nebo zdali obsahují nebo neobsahují úložiště elektrické energie. [1]

Z hlediska jednotlivých prvků se mikrosíť skládá z již výše zmíněných decentralizovaných zdrojů energie, a to jak malých elektráren, tak i různých druhů úložišť. Každý zdroj má buďto vlastní řídící jednotku nebo celý provoz mikrosítě zajišťuje centrální řídící jednotka případně se jedná o kombinaci obou řešení. Některé zdroje jsou standardně provozovány na maximální výkon, jiné jsou pomocí řídících jednotek provozovány podle potřeby.

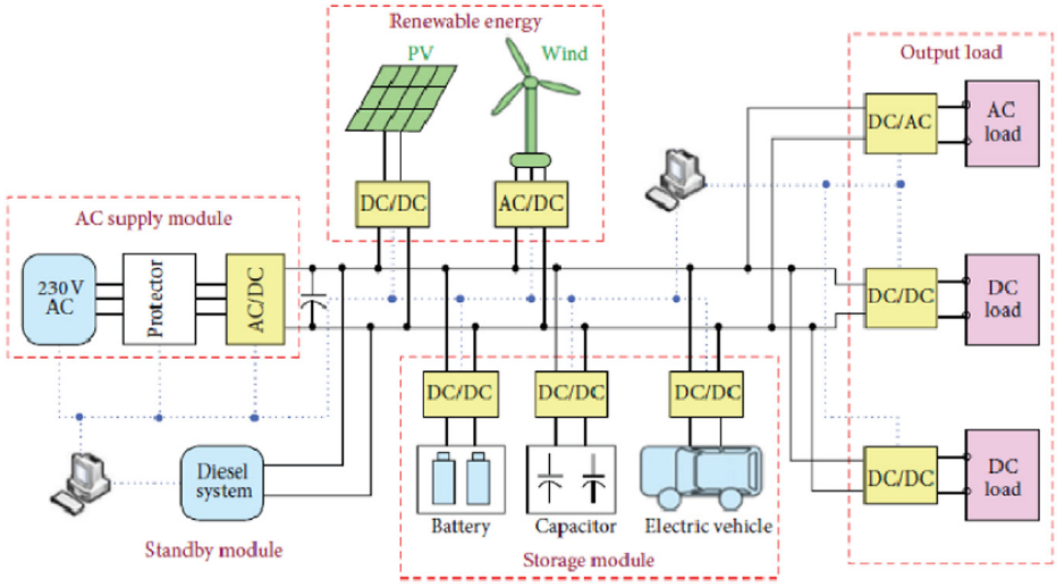


Obrázek 1.1: Architektura mikrosítě. [1]

1.2 Distribuční systémy

Distribuční systémy rozdělujeme u mikrosítí standardně podle typu napětí na stejnosměrné, střídavé a hybridní. Stejnosměrný přenos v mikrosítích je perspektivní obzvláště díky de-

centralizovaným zdrojům, které generují stejnosměrný proud (např. fotovoltaické panely). V takovém případě není potřeba střídač ke každému zdroji, ale stačí pouze jeden centrální měnič, čímž se redukují náklady i ztráty. Podle provedených studií [2] jsou i náklady na většinu komponentů stejnosměrné distribuční soustavy nižší (vodiče, izolace ad.) a taktéž jsou menší provozní náklady díky neexistenci jalového výkonu a skin efektu, menším úbytkům napětí, menšímu vlivu korony, a tedy celkově nižším ztrátám. Tyto výhody samozřejmě neplatí u konvenční elektrizační soustavy, která je založena na střídavých strojích a transformátorech. Nevýhodou je tedy malé množství zkušeností s DC vedením a nedostatečně rozvinuté technologické procesy s ním související, což znamená nižší spolehlivost a vyšší cenu některých komponent. Ukázka DC mikrosítě je na obr. 1.2.



Obrázek 1.2: Struktura DC mikrosítě. [1]

Standardní střídavé mikrosítě pracují na síťové frekvenci 50 Hz (nebo 60 Hz). Takovou mikrosíť lze tedy napojit přímo přes rozvaděč na distribuční síť bez potřeby měniče. Jelikož AC vedení naprosto převažuje v přenosových a distribučních sítích, jedná se o nejsnáze aplikovatelný způsob z hlediska propojení se zbytkem sítě a spotřebiteli.

Speciálním typem AC mikrosítí, který se díky některým výhodam testuje jsou vysokofrekvenční střídavé (HFAC) mikrosítě. Jejich výzkum byl však dosud pouze minimální. Pro zvýšení frekvence (obvykle na 500 Hz) z decentralizovaných zdrojů využívají frekvenční měniče. Před místem spotřeby je frekvence opět snížena na 50 Hz. Výhodou vyšší frekvence je např. snazší filtrace vyšších harmonických, snížení flickeru, zvýšení účinnosti zdrojů světla nebo snížení velikosti transformátorů, filtrů a dalších komponent. [3]

Ve většině případů je mikrosíť složena z různé kombinace decentralizovaných zdrojů, z

nichž některé produkují DC a některé AC. V takovém případě může být nejvhodnější vybudovat v mikrosítí oba typy vedení, na které budou připojeny příslušné skupiny zdrojů oproti konverzi měničem u každého zdroje. Výhody této konfigurace silně závisí na struktuře mikrosítě, proto je srovnání s ostatními typy poměrně obtížné.

1.3 Řízení mikrosítě

Pro ekonomický a spolehlivý provoz mikrosítě je třeba zvolit vhodné řízení. Základní požadavky pro řídící jednotky jsou následující:

- Regulace napětí a frekvence
- Efektivní sdílení zátěže (např. vyrovnávání činného a jalového výkonu a dostatečná komunikace mezi zdroji)
- Bezproblémový přechod mezi připojením/odpojením od sítě (Rychlá synchronizace a detekce ostrovního režimu)
- Ekonomické řízení provozu decentralizovaných zdrojů
- Řízení toku výkonu mezi mikrosítí a elektrizační soustavou

Těchto požadavků lze dosáhnout několika způsoby. Základní druhy řízení v mikrosítích jsou centralizované, decentralizované a distribuované. Každý z těchto způsobu najde své využití v závislosti na struktuře míkrosítě, použitých komponentech atd. [4]

1.3.1 Centralizované

Centralizované řízení se využívá většinou pro malé mikrosítě. Všechny informace o stavu jednotlivých komponent sítě jsou přenášeny z příslušných senzorů a zařízení pomocí komunikační sítě do centrální řídící jednotky. Jednotka přijaté signály zpracuje, vyhodnotí stav mikrosítě a v případě potřeby vydá příkazy dílčím řídícím jednotkám jednotlivých komponent přes stejnou komunikační síť. V případě mikrosítí připojených k elektrizační soustavě a systémů několika mikrosítí se využívá ještě další nadřazená řídící jednotka na úrovni distribuční soustavy. Celkem se tedy centralizované řízení může skládat ze tří úrovní řídících jednotek.

1.3.2 Decentralizované

Jak z názvu vyplývá, decentralizované mikrosítě nemají jednu centrální řídící jednotku. Každý komponent sítě má vlastní řídící jednotky s uloženými algoritmy, které provádí

veškeré rozhodování. Tyto jednotky se většinou dělí pouze na vyšší a nižší úrovně, jedná se tedy maximálně o dvouvrstvou architekturu.

Decentralizované řízení nalezne své využití v mikrosítích složených z mnoha prvků, kde by centrální řízení bylo příliš pomalé vzhledem k obrovskému množství informací zpracovávaných v jednom bodě komunikační sítě. Mezi další výhody oproti centralizovanému řízení patří jednodušší algoritmy, jednodušší připojení nového prvku do sítě, nízké požadavky na komunikační technologie, vyšší odolnost vůči poruchám (havárie jednoho komponentu nemá výrazný vliv na celou mikrosíť) a lepší flexibilita provozu. Nevýhodami je naopak horší koordinace mezi prvky sítě, většinou neoptimální řízení a neschopnost složitějšího provozu.

1.3.3 Distribuované

Distribuovaný způsob řízení je podobný decentralizovanému, kdy každý prvek má vlastní řídící jednotku, která je ale tentokrát schopná komunikovat s řídícími jednotkami sousedních prvků. Každá jednotka tedy stále řídí svůj prvek samostatně, ale vykonává rozhodnutí s využitím informací i od okolních prvků, čímž je dosaženo komplexnějšího provozu sítě. Tento způsob řízení kombinuje výhody obou přecházejících, proto je považován za nejperspektivnější.

Srovnání výhod a nevýhod jednotlivých metod řízení z heldiska různých vlastností naleznete v tabulce 1.1.

1.4 Využití umělé inteligence pro řízení mikrosítí

Umělá inteligence (Artificial Inteligence, AI) má v řízení mikrosítí a v budoucnosti i řízení celých distribučních a přenosových sítí velký potenciál. Nástroje umělé inteligence jsou schopné efektivně zpracovat extrémní množství dat, která jsou v chytrých sítích generována, a vyřešit vysoce komplexní problémy pro spolehlivý a bezpečný chod sítě. Pomocí nástrojů AI lze provádět současné způsoby řízení rychleji, efektivněji a přesněji, zároveň se algoritmy na bázi AI mohou bez zásahu člověka samostatně vyvíjet, anebo umožnit aplikaci nových pokročilejších způsobů řízení, na které současné konvenční metody nestačí. Nástrojů AI je ale mnoho a zabývat se všemi není vzhledem k rozsahu této práce možné, proto se v následujících podkapitolách budu zabývat pouze umělými neuronovými sítěmi, které jsou v současnosti nejpopulárnější a nejperspektivnější.

Tabulka 1.1: Srovnání způsobů řízení mikrosítí [5]

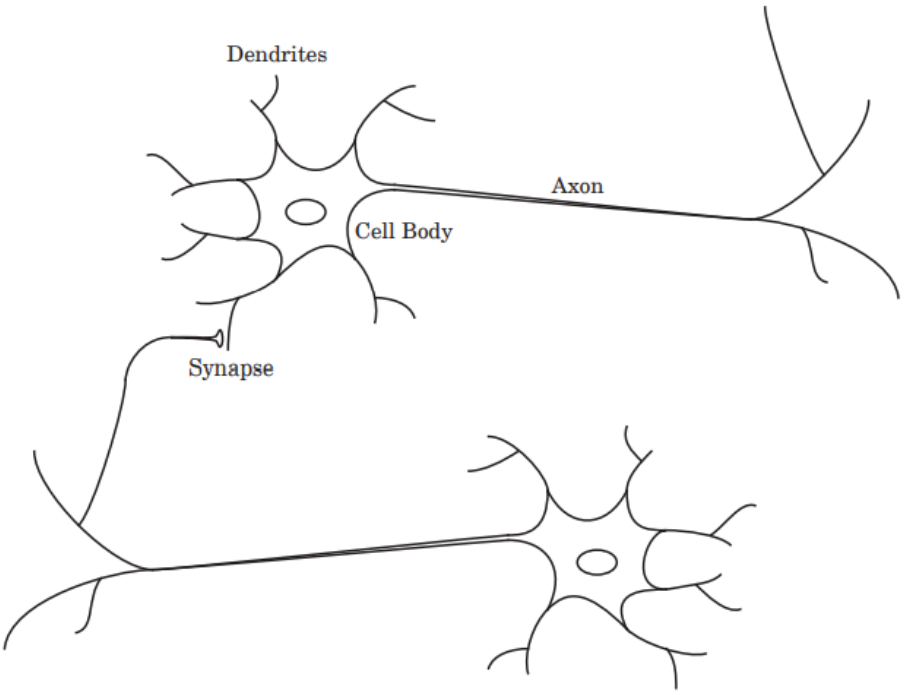
Prvek řízení	Centralizované	Decentralizované	Distribuované
Napěťová stabilita	Méně efektivní s více DER	Efektivní a realizovatelné	Vysoce efektivní, obtížné realizovatelné
Regulace frekvence	Méně efektivní s více DER	Efektivní a realizovatelné	Vysoce efektivní, obtížné realizovatelné
Řízení frekvence zátěže	Efektivní a realizovatelné	Méně efektivní, nemožnost celkového řízení	Vysoce efektivní, obtížné realizovatelné
Power-sharing	Méně efektivní s více DER	Efektivní a realizovatelné	Vysoce efektivní, obtížné realizovatelné
Optimální tok výkonu	Efektivní ale obtížné	Méně efektivní, nemožnost celkového řízení	Vysoce efektivní, obtížné realizovatelné
Energy management	Efektivní ale obtížné	Méně efektivní, nemožnost celkového řízení	Vysoce efektivní, obtížné realizovatelné
Řídící vrstvy	Jedna	Více	Více
Implementace	Jednoduchá	Průměrná	Komplexní
Cíle	Jeden explicitní úkol	Více úkolů	Proměnné a nejisté
Felxibilita	Nízká	Průměrná	Vysoká
Komunikace	Vysoká	Nízká	Střední
Spolehlivost [1]	Single point	Multiple points	Multiple points
Škálovatelnost	Nízká	Průměrná	Vysoká
Plug-and-play	Obtížné	Dosažitelné	Dosažitelné

1.4.1 Umělé neuronové sítě

Jak z názvu vyplývá, umělá neuronová síť (Artificial Neural Network, ANN) je nástroj umělé inteligence inspirovaný lidskou nervovou soustavou. Základními prvky nervové soustavy jsou nervové buňky zvané neurony. Neuron tvoří tři hlavní části: tělo buňky, dendrity a axon (viz obr. 1.3). Dendrity přijímají vstupní signály(nervové vzruchy) a přenášejí je do těla buňky. Tělo buňky tyto signály zpracuje a přes axony je může předat dalším neuronům. Kontakt mezi axony a dendrity dalšího neuronu se nazývá synapse. V průběhu života se nervová soustava vyvíjí, zpočátku se vytváří formují nová spojení, v pozdějších fázích života dochází převážně k zesilování a zeslabovaní synaptických kontaktů. Na podobném principu operují i umělé neuronové sítě, jejichž elementární prvky se stejně jako nervové buňky nazývají neurony. Tyto neurony jsou vzájemně propojeny, a právě tato spojení definují funkci nervové sítě. Každé spojení se totiž v průběhu učení neuronové sítě upravuje, dokud není schopna s požadovanou přesností aproximovat vztah mezi vstupními a výstupními hodnotami.

Je třeba dodat že biologické neurony jsou oproti umělým sice pomalejší, ale podstatně komplexnější. A vzhledem k tomu, že pracují všechny současně a nervová soustava jich obsahuje opravdu mnoho zatím nejsme schopni takovou síť vytvořit uměle.

V následujících kapitolách budou probrány typy struktur umělých neuronových sítí a možnosti jejich strojového učení.



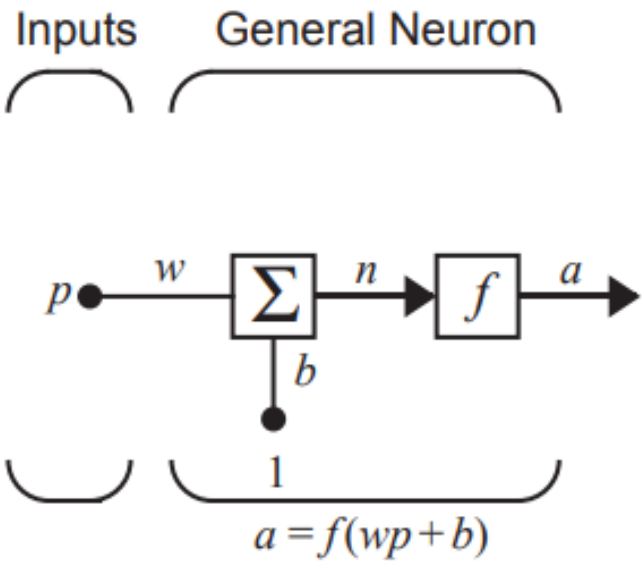
Obrázek 1.3: Schéma základní struktury biologického neuronu [6]

1.4.2 Architektura umělé neuronové sítě

Jak bylo již v úvodu zmíněno, základním stavebním blokem umělé neuronové sítě je neuron. Nejjednodušší uměla neuronová síť tedy obsahuje právě jeden neuron (viz obr. 1.4). Každý neuron musí přijímat nějaký skalární vstup, který budeme značit p. Před vstupem do neuronu je hodnota vstupu upravena přenásobením skalární vahou w. Uvnitř neuronu se k upravenému vstupu připočte tzv. bias b, který vstupní hodnotu podle potřeby vychýlí. Na výsledek této sumy, kterou budeme nazývat vážený vstup n, se aplikuje aktivační (též přenosové) funkce f, která vypočte konečný výstup neuronu a. Vzorec pro tuto neuronovou síť tedy vypadá následovně:

$$a = f\left(wp + b\right). \tag{1.1}$$

Váha a bias neuronu jsou parametry, které se v průběhu učení neuronové sítě upravují pro dosažení požadovaného výsledku. Váhu obsahuje neuron vždy, bias lze v případě nadbytečnosti vynechat. Avšak je potřeba vzít v úvahu, že při absenci biasu projde nulová vstupní hodnota přes sčítačku beze změny. Aktivační funkce je statická a volí se při návrhu. Lze si všimnout, že umělý neuron opravdu připomíná svou funkcí neuron biologický. Váha odpovídá síle synaptických kontaktů, sumační prvek s aktivační funkcí pracuje obdobně jako tělo neuronu a výstupem neuronu je signál putující axonem.



Obrázek 1.4: Nejjednoduší případ ANN [6]

Druhy aktivačních funkcí

Aktivačních funkcí je mnoho. Může se jednat o jakoukoli lineární nebo nelineární matematickou funkci, která vhodně modeluje očekávaný vztah mezi vstupy a výstupy. Vybrané často používané aktivační funkce s jejich anglickými názvy naleznete v tabulce 1.2. Grafické vyjádření bylo převzato z programu Matlab a obsahuje proto i příkaz pro danou funkci.

Jednou z nejjednodušších je aktivační funkce s pevnou limitou (Hard limit), která převádí vstup pouze na nuly a jedničky. V případě, že je na vstupu záporné číslo, bude na výstupu nula. Pokud je vstup kladný nebo roven nule, bude na výstupu jednička. Taková funkce nalezne využití, potřebujeme-li klasifikovat vstupy do dvou kategorií. Důležité je poznamenat, že vstupem aktivační funkce je již transformovaný vstup pomocí vah a biasu, tudíž se nejedná o pouhou třídičku kladných a záporných čísel.

Pokud nechceme, aby aktivační funkce ovlivňovala hodnotu výstupu, použijeme lineární funkci, která zobrazí na výstupu vstupní hodnotu. Veškeré změny vstupních hodnot jsou tak prováděny pouze úpravou vah a biasu. Tato funkce je například využita ve výchozí mělké neuronové sítí v programu Matlab.

Komplexnější, ale taktéž často používané funkce jsou tzv. Log-Sigmoid a Tan-Sigmoid funkce. Log-sigmoid redukuje pomocí exponenciály vstup do intervalu mezi 0 a 1. Tan-Sigmoid je hyperbolický tangens, a tedy redukuje vstupní hodnoty na interval -1 až 1. Tuto funkci lze opět nalézt ve výchozím nastavení mělké neuronové sítě v programu Matlab.

Obě funkce jsou ti velmi podobné, avšak Tan-Sigmoid má například oproti Log-Sigmoid vyšší gradient, tudíž v jednom kroku provádí výraznější úpravy vstupních hodnot. Díky tomu a zároveň své symetrii kolem nuly dosahuje rychlejší konvergence.

Komplexnější struktury neuronových sítí

Struktura s pouze jedním neuronem je vhodná na přehledné představení všech základních funkčních prvků matematického modelu umělé neuronové sítě, ale v praxi přílišné využití nenajde. Většina reálných aplikací totiž pracuje s více než jedním vstupem a vyžaduje komplexnější funkce, na které je třeba preciznější a jemnější ladění přepočtových parametrů. V takových případech se proto použije síť složená z několika neuronů. Do každého neuronu vstupují všechny vstupní hodnoty ve formě vektoru $\mathbf{P} = (p_1, p_2, ..., p_R)$, kde R je počet vstupů. Ke každé vstupní hodnotě je přiřazena vlastní váha $w_{i,j}$, kde i značí index neuronu a j značí index vstupu. Celkový počet neuronu v sítí budeme značit S. Každý neuron má navíc vlastní bias b_i, tudíž pro jeden neuron lze napsat rovnici:

$$\mathbf{a} = f\left(\mathbf{wp} + \mathbf{b}_i\right), \tag{1.2}$$

kde $\mathbf{w}$ je vektor vah $(w_{1,1}, w_{1,2}, ..., w_{1,R})$.

Pro celou síť pak platí:

$$\mathbf{a} = f\left(\mathbf{Wp} + \mathbf{b}\right), \tag{1.3}$$

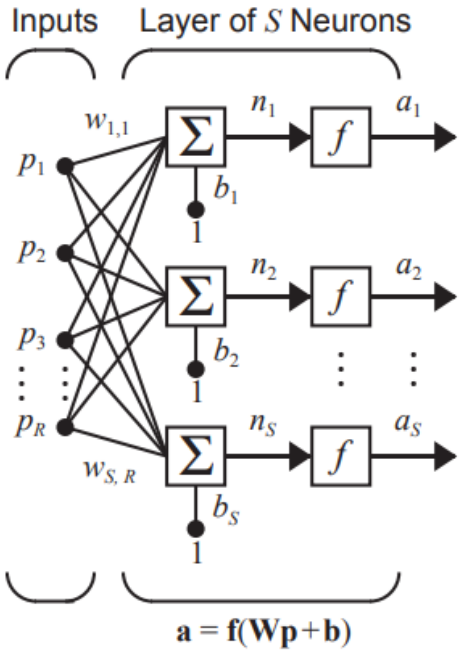
kde $\mathbf{a} = (a_1, a_2, ..., a_S)$ je vektor výstupních hodnot neuronů , b je vektor biasů a W je matice vah ve tvaru:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & ... & w_{1,R} \\ w_{2,1} & w_{2,2} & ... & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,1} & ... & w_{S,R} \end{bmatrix}$$

Schéma sítě popsané těmito vztahy je znázorněno na obr. 1.5.

Jak bylo již zmíněno, každý neuron přijímá všechny vstupy, pro každou kombinaci vstupu a neuronu existuje jedna váha a všechny neurony mají jeden bias. Takové schéma je ale velmi nepřehledné, zakreslujeme-li ho stejným způsobem jako v případě elementární struktury o jednom neuronu. Proto se jednotlivé části schéma redukují do bloků. Výsledné blokové znázornění neuronové sítě naleznete na obr. 1.6.

Tuto strukturu označujeme jako vrstvu. Pokud chceme analyzovat ještě komplexnější vztahy můžeme na výstup této vrstvy připojit další. Výstupní vektor $\mathbf{a}$ tak bude sloužit



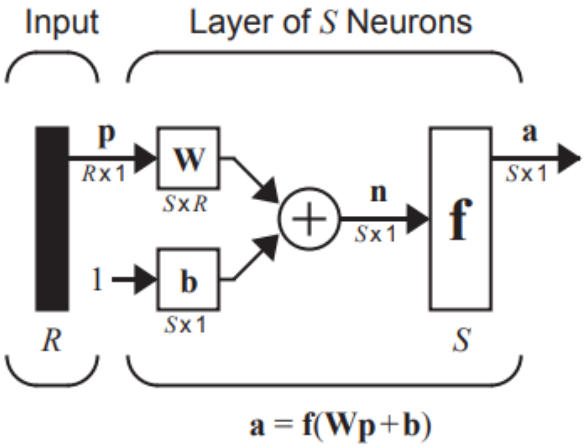
Obrázek 1.5: Schéma umělé neuronové sítě s R vstupy a S neurony. [6]

jako vstupní vektor pro další totožnou strukturu a vzniká vícevrstvá neuronová síť (Multilayer Neural Networks, MLNN). Poslední vrstva této sítě se nazývá výstupní, všechny předcházející se nazývají skryté vrstvy. Často je od ostatních odlišována i první vrstva a označována jako vstupní. Skryté vrstvy, kterými prochází vstupní hodnoty nejprve popisujeme jako nižší, následující vrstvy jako vyšší. V nižších vrstvách se nejprve analyzují nejjednodušší prvky analyzované problematiky, ve vyšších vrstvách se postupně zvyšuje komplexita. Počet vrstev má podstatný vliv na funkci neuronové sítě a zdaleka ne vždy vede použití většího množství k lepším výsledkům. Názorným příkladem je digitální zpracování obrazu. V nejnižších vrstvách se rozpoznají rohy, hrany, zatímco v nejvyšší vrstvě může být systém schopen rozpoznat celý objekt. Práce s takovými ANN se nazývá hluboké učení (Deep Learning, DL). [7]

Chceme-li nakreslit vícevrstvou neuronovou síť schematicky, postačí zapojit za sebou několik blokových schémat z obr. 1.6 a přidat dodatečné indexování pro rozlišení jednotlivých vrstev. Matematicky by se například třívrstvá neuronová síť zapsala takto:

$$\mathbf{a}^3 = f^3\left(\mathbf{W}^3 f^2\left(\mathbf{W}^2 f^1\left(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1\right) + \mathbf{b}^2\right) + \mathbf{b}^3\right). \tag{1.4}$$

Lze si všimnout, že u všech dosud zmíněných struktur cestuje informace pouze jedním směrem od vstupu k výstupu. To znamená, že neurony nedostávají žádnou zpětnou vazbu o svých výsledcích a žádná ze skrytých vrstev nemůže ovlivnit vrstvu přecházející. Takovému typu umělé neuronové sítě se říká dopředná (Feed forward). Chceme-li, aby se



Obrázek 1.6: Blokové schéma umělé neuronové sítě s R vstupy a S neurony. [6]

informace o výstupu autonomně dostala do skrytých vrstev, musíme použít síť umožňující obousměrný tok informací tzv. rekurentní (Recurrent nebo též feedback). Takto je možné srovnat získané výstupní hodnoty s požadovanými a okamžitě poslat informaci o odchylce zpět do skryté vrstvy, kde neurony přepočítají hodnoty vah a biasu a výsledky odešlou neuronům v přecházející vrstvě. Proces se opakuje, dokud není dosaženo vstupní vrstvy. Síť se tak sama dynamicky vyvíjí i bez změny vstupních dat, díky čemuž je vhodná její aplikace v časově proměnlivých procesech. Ukázka jedné rekurentní neuronové sítě je na obr. 1.7. Tato síť obsahuje navíc blok zpoždění, který zpozdí jeho vstup $\mathbf{u}$ o jeden časový krok. Vstupem do této sítě jsou pouze počáteční podmínky $\mathbf{a}(0)$ a všechny následující vstupy jsou již předcházející výstupy. Pro funkci zpožďovací bloku platí:

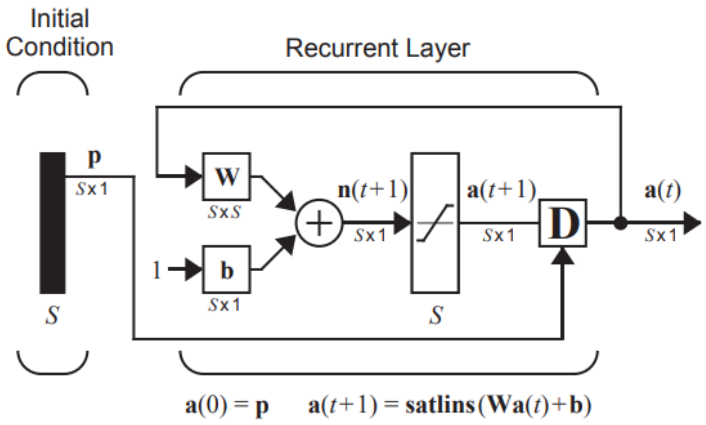
$$\mathbf{a}(t) = \mathbf{u}\,(t-1)\,, \tag{1.5}$$

Dalším častým blokem v rekurentních sítích je integrátor, který přepočítává svůj vstup podle vztahu:

$$\mathbf{a}(t) = \int_0^t \mathbf{u}(\tau)\,d\tau + \mathbf{a}(0). \tag{1.6}$$

Typy učení neuronových sítí

Z hlediska trénování ANN rozlišujeme tři typy učení: s učitelem (supervised learning, SL), bez učitele (unsupervised learning, UL) a zpětnovazební učení (reinforcement learning, RL). Při SL vkládáme do sítě pevně daný vstup a výstup a požadujeme nalezení výstupu s co nejmenší odchylkou od zadaného. UL se využívá v případě, že neznáme výstupy a chceme pouze lépe zorganizovat vstupní data nebo v nich nalézt zatím neznámé souvislosti.



Obrázek 1.7: Příklad rekurentní neuronové sítě [6]

V současnosti velmi populárním algoritmem při řešení mnoha problémů je RL. RL se skládá ze čtyř hlavních částí: prostředí, agent, akce a odměna. Agent se nachází v nějakém prostředí a provádí akce. Za každou akci je buďto odměněn nebo potrestán. Každá odměna i trest má různou váhu. Cílem agenta je většinou dosáhnout maximální kumulativní odměny. Agent tedy musí uvažovat v dlouhodobém měřítku, ne se snažit pouze o maximalizaci odměny za každou jednotlivou akci. Pro formální zápis problematiky lze využít Markovův rozhodovací proces (Markov decision process, MDP). Základem MDP jsou v tomto případě čtyři parametry: množina stavů $S(\forall s \in S)$, množina akcí $A(\forall a \in A)$, přechodová funkce $\tau : S \times A \times S \rightarrow [0, 1]$, která udává pravděpodobnost, že provedením akce a ve stavu s v čase t přejde systém do stavu s' v čase $t + 1$, a funkce odměny $\Re : S \times A \times S \rightarrow \mathbb{R}$, jejíž výstupem je okamžitá odměna, kterou agent obdrží po přechodu ze stavu s do s'. Pro pravděpodobnost přechodu platí $p_a(s, s') = p(s_{t+1} = s' | s_t = s, a_t = a)$. K řešení takto definované problematiky RL se využívá více způsobu, nejčastěji se jedná o Q-learning a SARSA (State-Action-Reward-State-Action). Q-learning je algoritmus, který před začátkem učení vypočte výchozí hodnotu tzv. kvalitu kombinace stavu a akce. Pro každý další stav se vypočte tzv. Q-value podle vztahu

$$Q^{\pi}(s, a) = r(s, a) + \gamma \sum_{s'}^{0} p(s'|s, a) V^{\pi}(s'). \tag{1.7}$$

Vztah popisuje situaci, kdy je agent ve stavu s a provede akci a, za což obdrží okamžitou odměnu $r(s, a)$. Poté agent dodržuje určitou strategii π, ze které vychází hodnoty následujících stavů $V^{\pi}(s')$. Celý systém se v čase s určitou pravděpodobností mění, což popisuje funkce $p(s|s, a)$ a odměny ztrácejí na hodnotě v závislosti na velikosti diskontu $\gamma \in [0, 1]$. Q-value je vypočtena pro každý pár stav-akce, a při každém opakovaní algoritmu se tyto hodnoty přepočtou, dokud není nalezena optimální strategie. Optimální strategie znamená získání maximální možné diskontované budoucí odměny. SARSA je

založena na stejném principu jako Q-learning, akorát nemění Q-values vždy pouze na základě maximální odměny.[8] [9]
[10]

1.5 Hierarchické řízení mikrosítí

V přecházející sekci byly vysvětleny vybrané nástroje umělé inteligence. Nyní se podíváme na jejich aplikaci u vybraných typů řízení. V úvodním rozdělení byly ukázány druhy řízení podle umístění řídících jednotek. Pro ukázku schopností nástrojů umělé inteligence však bude vhodnější zaměřit se na specifické typy řízení z hlediska jejich funkce a rychlosti. Takové rozdělení řízení se nazývá hierarchické.

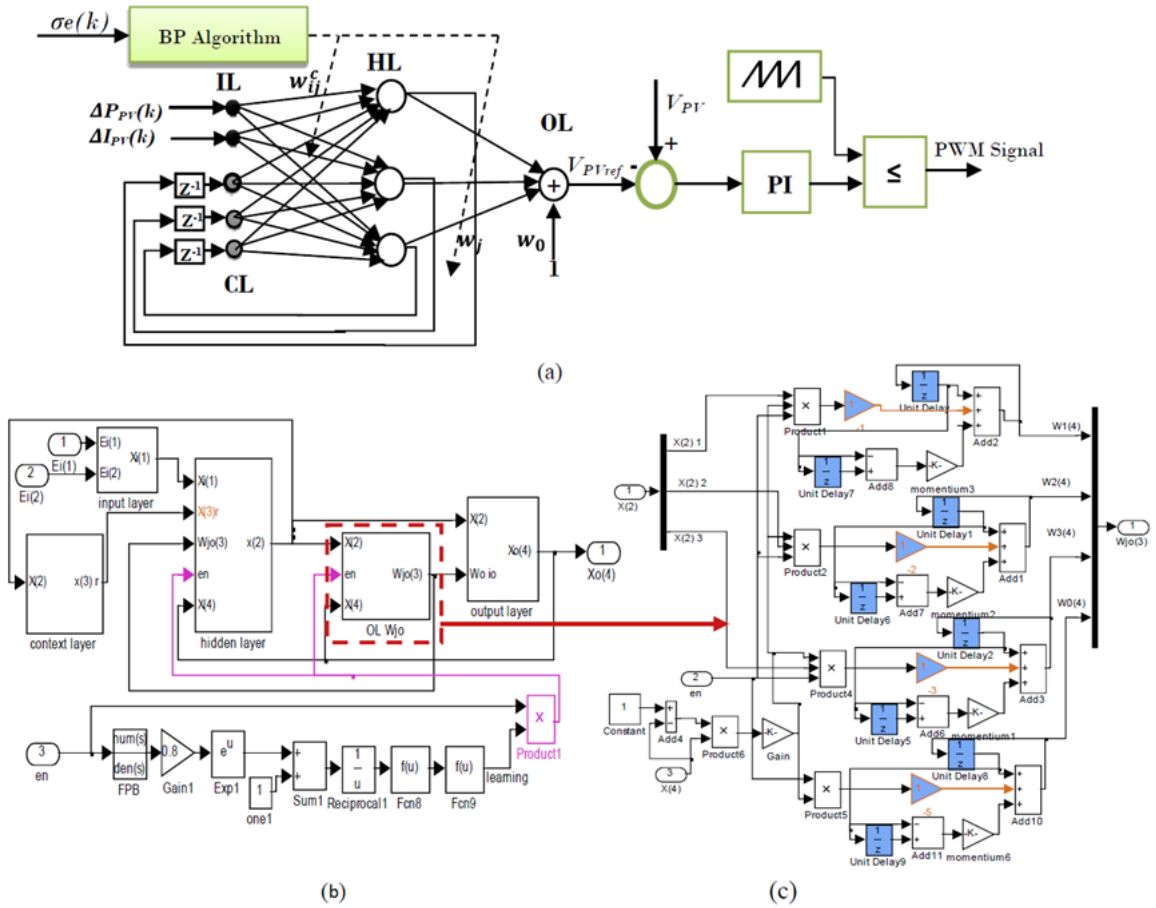
Hierarchické řízení je založeno na rozdílu mezi časovými požadavky řízení různých parametrů. Podle rychlosti řízení je rozděleno na primární (Automatická regulace frekvence), sekundární (Regulace výkonové rovnováhy s automatickou aktivací) a terciární (Regulace výkonové rovnováhy s manuální aktivací). Primární se využívá pro okamžité řízení prvků mikrosítě zpravidla v rámci milisekund až sekund. Sekundární stabilizuje odchylky ustáleného stavu vyvolané primárním řízením řadově v minutách. Terciární je nejpomalejší (minuty až hodiny) a slouží k řízení toků výkonu mezi nadřazenou soustavou a mikrosítí.

1.5.1 Automatická regulace frekvence

Cílem automatické regulace frekvence je řídit výkonové toky každého decentralizovaného zdroje kdykoli se změní zátěž nebo dojde k přerušení provozu některého ze zdrojů, a to během maximálně několika sekund. Regulací výkonových toků se redukuje odchylka síťové frekvence a udržuje stabilita mikrosítě. To zprostředkovávají lokální řídící jednotky, do kterých přes komunikační síť vstupují informace o ustálených hodnotách výkonu a napětí.

V mikrosítích se automatická regulace frekvence převážně na okamžité sdílení výkonu, sledování bodu maximálního výkonu (Maximum power point tracking, MPPT) a řízení setrvačnosti. Studie ukazují [11], že MPPT systémy s konvenčními metodami jsou pomalé a vyžadují komplexní návrh, zatímco metody založené na umělých neuronových sítích jsou schopné sledovat bod maximálního výkonu s chybou menší než 0,1 %. Jedním takovým příkladem je studie [12] zabývající se řízením MPPT pole fotovoltaických modulů pomocí Elmanovy neuronové sítě (Elman neural network, ENN) a PI regulátoru. V případě tohoto přístupu vstupuje do neuronové sítě změna výstupního proudu modulů a odchylka výkonu, výstupem ENN je referenční napětí modulů. Srovnáním této hodnoty s naměřenou se získá

odchylka, která vstupuje do PI regulátoru, který vygeneruje PWM signál pro DC/DC boost měnič. Struktura použité ENN je na obr. 1.8. Jedná se o dynamickou rekurentní umělou neuronovou síť složenou ze čtyř vrstev: vstupní, skryté, kontextové a výstupní. Kontextová vrstva slouží k dočasnému zapamatování informací o přecházejících stavech neuronů skryté vrstvy, které jim posílá zpátky. Výsledkem této struktury bylo dosažení střední kvadratické chyby sledování pouze 0,0004 a dosažení ustáleného stavu trvalo 4 ms s minimálními fluktuacemi. Nejednalo se však o simulaci s proměnlivými okolními podmínkami. Tím se zabývala studie [13], kde bylo analyzováno přizpůsobení neuronové sítě (stejný typ jako v předcházející studii) změnám ozářenosti a rychlosti větru. V takovém případě trvalo nalezení bodu maximálního výkonu 38 ms, což je podstatně rychlejší než standardně užívaný algoritmus inkrementální vodivosti, který dosahuje MPP za 92 s.



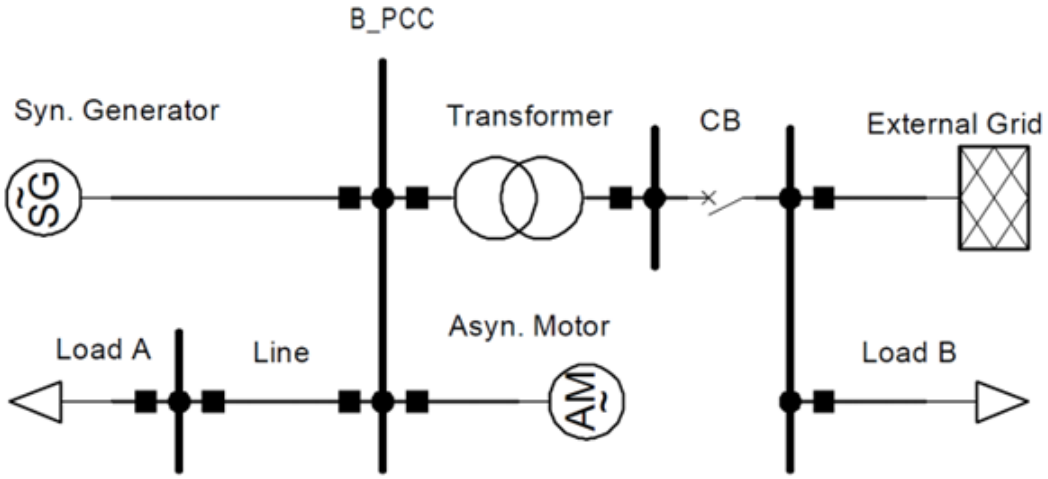
Obrázek 1.8: Ukázka MPPT pomocí Elmanovy neuronové sítě - a) Struktura ENN, b) Struktura ENN v Simulinku, c) Příklad bloku pro adaptaci neuronových vah [12]

1.5.2 Regulace výkonové rovnováhy s automatickou aktivací

Regulace výkonové rovnováhy s automatickou aktivací převážně slouží k redukci odchylek způsobených automatickou regulací frekvence během několika minut od vzniku. V

případě konvenčních metod je tato metoda regulace v mikrosítích pomalá, nepřesná a vyžaduje rozsáhlou komunikační infrastrukturu. Nedostatečná komunikace totiž způsobuje problémy se synchronizací zdrojů a udržením stability.

Nové technologie na bázi umělé inteligence mohou tyto nedostatky vyřešit, avšak zatím nebylo dosaženo zcela uspokojivých výsledků. Velké množství úložišť energie potřebných pro doplnění provozu obnovitelných zdrojů výrazně komplikuje požadavky na řízení. Většina studií se proto zatím zabývala pouze omezenou strukturou mikrosítě s nízkým počtem a variací zdrojů, úložišť a zátěží, a často pouze v ostrovním režimu. Výsledky těchto studií jsou však poměrně slibné. Například studie [14] srovnala řízení napětí a frekvence pomocí PID regulátoru oproti systému s neuronovou sítí v jednoduché mikrosíti (viz obr. 1.9). Systém s neuronovou sítí dosáhl rychlejší reakce na dynamické změny v soustavě, menší výchylky při přeregulaci a kratší dobu ustálení. Střední kvadratická odchylka je o 40-50 % menší než u PID regulátoru.



Obrázek 1.9: Model jednoduché mikrosítě [14]

Poměrně perspektivním způsobem strojového učení pro automatickou regulaci výkonové rovnováhy je RL. Studie [15] se zabývala využitím RL pro kompenzaci jalového výkonu, nesymetrie proudů zátěže a odstraněním vyšších harmonických. Cílem řídícího bloku na bázi RL je generovat signál pro řízení zařízení DSTATCOM (Distributed static compensator). Funkcí DSTATCOMu je řídit tok výkonů a udržovat dynamickou stabilitu pomocí změny napětí a fázového úhlu napětí. K tomu využívá řízený napěťový měnič VSC (Voltage Source Converter) napájený stejnosměrným zdrojem (většinou baterie). VSC je šestipulzní můstkový měnič s IGBT tranzistory, který převádí vstupní DC napětí na třífázové výstupní AC napětí. K síti je připojen paralelně přes transformátor. [16] Uvažovaný způsob řízení v této studii umožňuje na základě lokálního napětí a proudu DSTATCOMu generovat signál pro řízení napětí a frekvence soustavy. Zároveň je řízeno

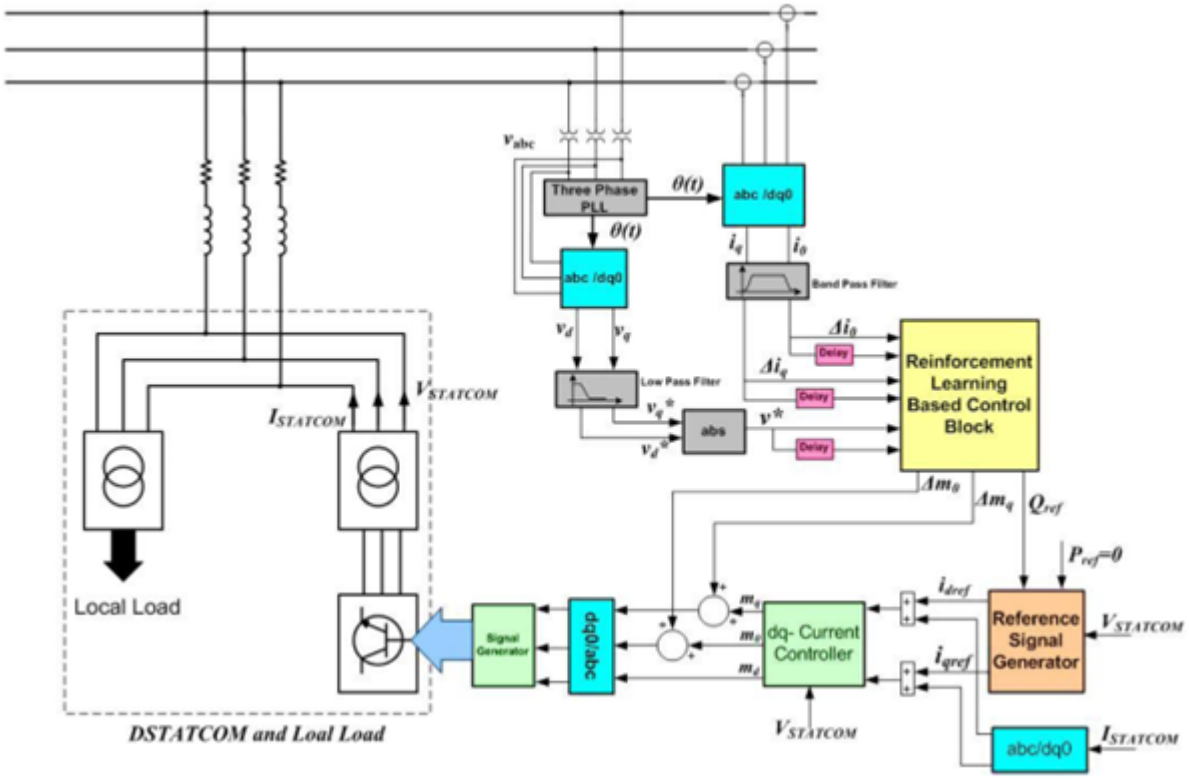
i napětí na PCC (Point of Common Coupling – místo připojení mikrosítě k distribuční síti) a toky výkonů decentralizovaných zdrojů. Struktura řídící logiky je znázorněna na obr. 1.10 a lze ji rozdělit na napěťovou a proudovou část. Napěťová část porovnává napětí PCC s referenčním a výsledná odchylka vstupuje do RL řídící jednotky, která vygeneruje referenční hodnotu jalového výkonu. V případě proudového řízení jsou odebírány proudy decentralizovaných zdrojů, které taktéž vstupují do RL řídící jednotky, která na základě jejich hodnot generuje korektivní signály pro redukci odchylek proudu. Veškeré řízení je provozováno online. Pro analýzu chování tohoto systému bylo simulováno nesymetrické zatížení, nelineární zatížení a třífázové zkraty. Výsledky simulace sice potvrzují správnost funkce řídícího algoritmu, ale chybí zde srovnání se současnými technologiemi.

Vzhledem k požadované vysoké kvalitě komunikace v oblasti automatické regulace výkonové rovnováhy můžeme nalézt i mnoho výzkumů zabývajících aplikací umělé inteligence právě v této problematice. Hlavním cílem je kompenzace zpoždění a nedokonalosti komunikace. Navržené modely jsou schopné doplnit neucelené informace o stavu na základě naučených prediktivních algoritmů v některých případech dokonce i se ztrátou až 60 % dat [17]. Samozřejmostí těchto modelů je také důraz na vysokou odolnost vůči kybernetickým útokům zejména z hlediska detekce injekce falešných dat.

Poslední důležitou záležitostí, která bude v této části zmíněna, je ochrana před poruchami, jejich případná detekce a oprava s následným uvedením mikrosítě zpět do provozu. Díky nekonzistenci výroby obnovitelných zdrojů je nestabilita sítě podstatným problémem. Zkraty v síti mohou způsobit oscilace v celém systému, což vede k fluktuacím napětí. Jedním ze způsobů detekce lokace zkratu je využití SVM (Support Vector Machine). Jakmile je detekována chyba, injektuje nejbližší zdroj vyšší harmonickou o vhodně zvolené frekvenci a v daném bodě se změří impedance. Informace o impedanci je zpracována SVM algoritmem, který určí lokaci poruchy s chybou do 0,25 %. Podobné modely na bázi neuronových sítí pracují ještě lépe a nevykazují téměř žádnou chybu. [18]

1.5.3 Regulace výkonové rovnováhy s manuální aktivací

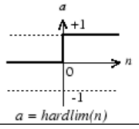
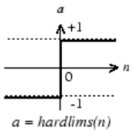
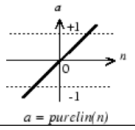
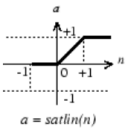
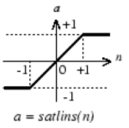
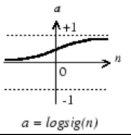
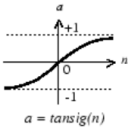
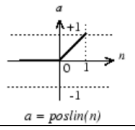
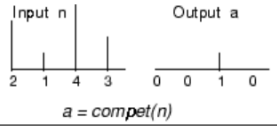
Manuálně aktivovaná regulace výkonové rovnováhy pracuje na úrovni distribuční soustavy. Zajišťuje optimální tok v mikrosítí a mezi mikrosítí a vnější soustavou. Mimo technické požadavky pro zajištění stabilního chodu sítě se zde projevují i ekonomické požadavky na co nejnižší provozní náklady. Pod tento typ regulace tedy spadají i operace související s energetickým trhem, mezi které patří například zprostředkování podpůrných služeb pro ostatní mikrosítě nebo vnější sítě v dosahu.



Obrázek 1.10: Příklad struktury řízení DSTATCOM v mikrosíti [15]

Základní problematikou této metody regulace je hledání optimálního toku výkonu a optimálního ekonomického provozu sítě s určitým omezením v podobě požadavku na kvalitu elektrické energie nebo mezní parametry provozu. Pro některé DC mikrosítě může být tato optimalizace konvexní, tedy existuje pouze jedno lokální, a proto i zároveň globální minimum. V případě AC mikrosítí se ale jedná o nekonvexní optimalizační problém, tudíž existuje více lokálních extrémů, ke kterým mohou použité algoritmy konvergovat a nedosáhnout tak hledaného globálního minima. Pro vytvoření algoritmu, který globálního minima dosáhne, je nutné velký výpočetní výkon a dostatečně komplexní model. Cílů optimalizace je několik. Může se jednat například o dosažení maximálního zisku provozovatele sítě, minimalizace provozních nákladů, minimalizace importu z vnější sítě, minimalizace doby provozu fosilních zdrojů energie, maximalizace využití obnovitelných zdrojů energie atd. Většinou se jedná o opravdu komplexní problémy, které se zabývají řízením různé kombinace zdrojů a úložišť. K optimálnímu provozu mikrosítě je totiž potřeba predikovat různé parametry ovlivňující výkon obnovitelných zdrojů a na základě těchto predikcí společně s predikcí spotřeby vhodně plánovat jejich provoz a taktéž provoz úložišť energie. K řešení této problematiky se využívají všechny již výše zmíněné nástroje AI.

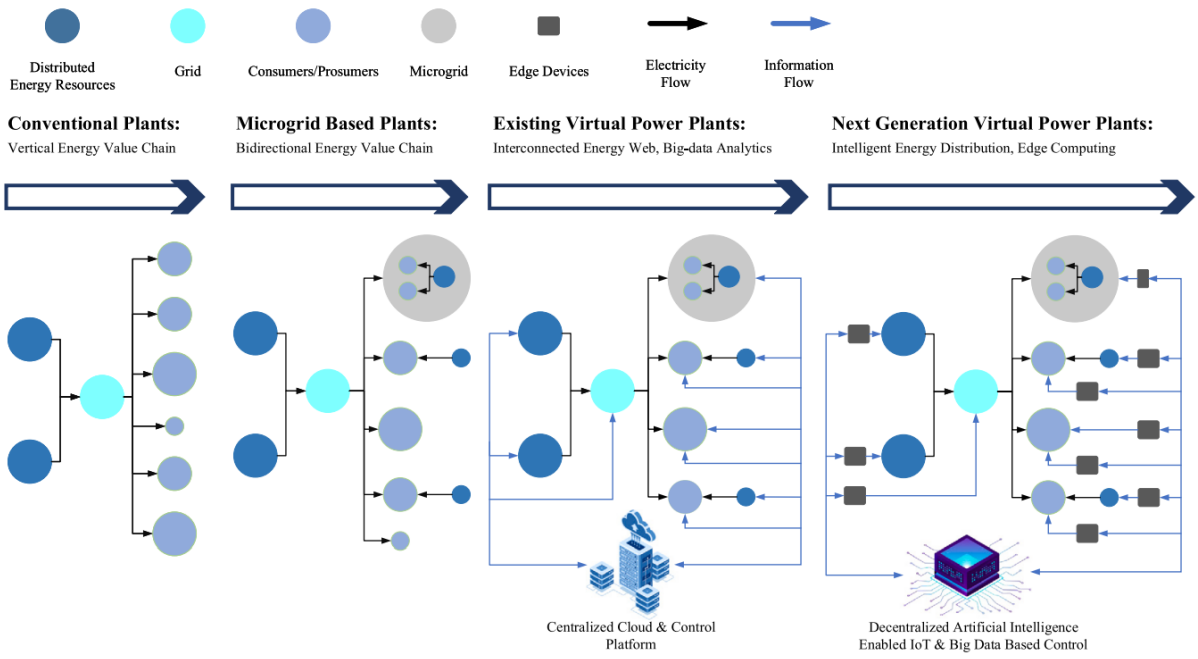
Tabulka 1.2: Seznam vybraných aktivačních funkcí [6]

Název	**Matematický zápis**	**Grafické vyjádření**
Hard limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$	
Symmetrical hard limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$	
Linear	$a = n$	
Saturating linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 0$ $a = 1 \quad n \geq 1$	
Symmetric saturating linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n \geq 1$	
Log-sigmoid	$a = \frac{1}{1+e^{-n}}$	
Hyperbolic tangent sigmoid	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$	
Positive linear	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$	
Competitive	$a = 1$ neuron s max n $a = 0$ ostatní neurony	

Kapitola 2

Virtuální elektrárny

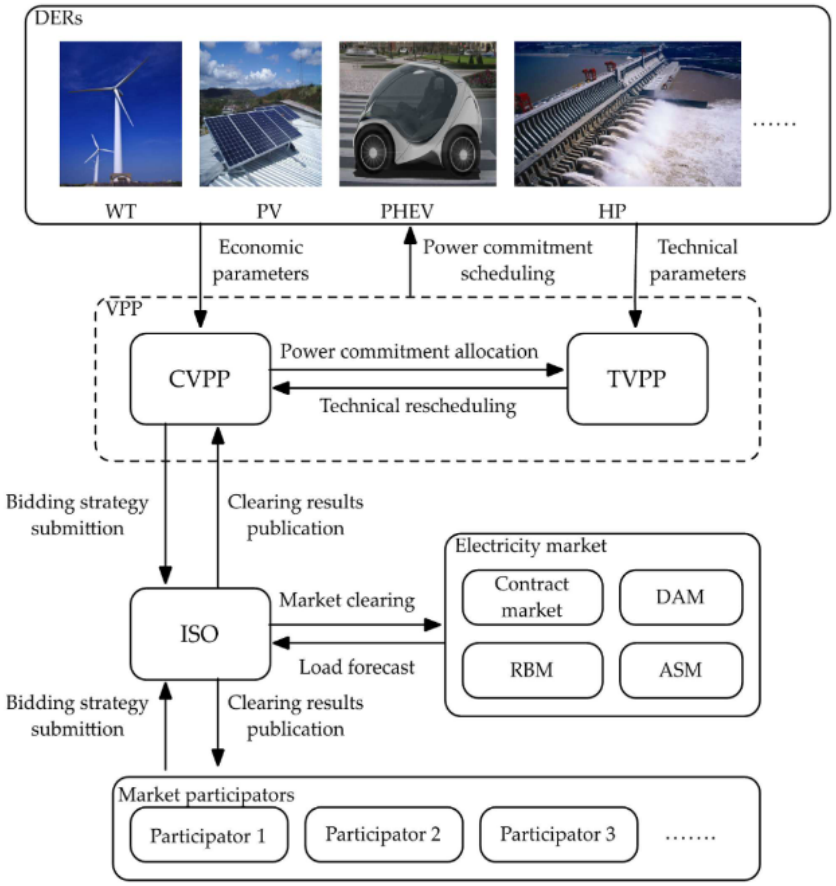
Definice virtuální elektrárny (Virtual Power Plant, VPP) není pevně stanovena, nejlépe lze pravděpodobně VPP specifikovat jako „inteligentnější“ a komplexnější mikrosíť, jejíž prvky nemusí být fyzicky spojeny. Hlavní rozdíly jsou naznačeny na obr. 2.1. Současné architektury virtuálních elektráren uvažují jednu centrální řídící platformu, která bezdrátově sbírá data, komunikuje a řídí všechny prvky soustavy. Nová generace virtuálních elektráren předpokládá ještě komplexnější strukturu využívající tzv. edge computing, kdy jsou výpočetní jednotky umístěny co nejblíže zdrojům dat (prvkům soustavy) pro minimalizaci časové prodlevy. Komunikace mezi všemi výpočetními jednotkami je zprostředkována pomocí IoT (Internet of Things) a rozsáhlé decentralizované softwarové struktury na bázi umělé inteligence obsahující obrovskou databázi. [19]



Obrázek 2.1: Srovnání struktury konvenčních elektráren, mikrosítí a virtuálních elektráren [19]

Virtuální elektrárnu lze rozdělit na dvě hlavní části: komerční (CVPP) a technická (TVPP). Hlavním cílem komerční virtuální elektrárny je ekonomická optimalizace. Zabývá se analýzou rizik, provozních nákladů a zisku za dodávku energie a nabídku služeb chytré sítě. Základní princip operace CVPP je pro všechny víceméně totožný. Nejprve jsou ekonomicky zhodnoceny všechny zdroje v rámci virtuální elektrárny, na základě čehož je každému přiřazena priorita. Poté CVPP nasbírá informace o trhu (požadavky spotřeby, historické strategie dalších účastníku trhu atd.) a vyprodukuje optimální profil generace pro každý zdroj. Poté odešle operátorovi trhu svou bidding strategii, který provede vyčištění trhu. Podle výsledků vyčištění je profil plánované generace optimalizován a finální verze je odeslána operátorovi pro bezpečnostní kontrolu. V případě porušení předpisu je plán přehodnocen a vygenerován nový. Mimo interakci s operátorem trhu se navrhované modely zabývají i různými dohodami mezi odběrateli a vlastníky decentralizovaných zdrojů buďto v rámci jedné virtuální elektrárny nebo mezi různými virtuálními elektrárnami případně uzavřením spolupráce s distributory.

Technická virtuální elektrárna se zabývá řešením komplexních technických problémů v oblasti spolehlivého a optimálního provozu sítě. Tyto problémy jsou velmi podobné jako v případě mikrosítí, tudíž se jedná o optimalizaci toku výkonů, komunikační protokoly, analýzu technické proveditelnosti navrhovaných řešení a zajištění rovnováhy mezi spotřebou a výrobou. Mezi důležité podkategorie patří opět detekce poruch a bezpečnost komunikačního prostředí. Základní postup operace TVPP je následující: TVPP nejprve shromáždí technické parametry integrovaných zdrojů včetně statických a dynamických provozních dat. Zjištěné informace předá operátorovi a obdrží bezpečnostní regulatorní požadavky (např. meze toků výkonu). Na základě těchto požadavků a ekonomického profilu CVPP provede TVPP bezpečnostní kontrolu a společně s CVPP případně vytvoří nutné modifikace. Jestliže nedostatky detekuje operátor, zajistí TVPP okamžitou nápravu a odložení provozu VPP, dokud není z hlediska bezpečnosti vše v pořádku. Při provozu sleduje TVPP stav všech zdrojů a zajišťuje optimální toky výkonů. Na konci daného cyklu vypočítá přesné výstupy všech zdrojů a předá je CVPP pro výpočet zisku. Graficky jsou vztahy mezi jednotlivými částmi VPP a dalšími účastníky trhu znázorněny na obr. 2.2. Trh s elektřinou na obrázku se skládá z trhu s kontrakty (Contract market), vyrovnávacího trhu, který koriguje odchylky od předpokládané spotřeby a výroby (Real-time balancing market, RBM), denních trhu (Day-ahead market, DAM) a trhu s podpůrnými službami (Ancillary service market, ASM). [20][21][22]



Obrázek 2.2: Struktura operace VPP [22]

2.1 Smart inverters

Jedním z nejdůležitějších prvků VPP jsou „chytré“ střídače (Smart Inverters, SI). Původně byl tento název používán pro střídače u fotovoltaických systémů s nadstandardními funkcemi (např. kompenzace jalového výkonu). Dnes jsou takto označovány střídače u všech decentralizovaných zdrojů s komunikačními a pokročilými řídícími funkcemi. V základu se tyto funkce dělí podle toho, kdo je vykonává a jaký je jejich účel.

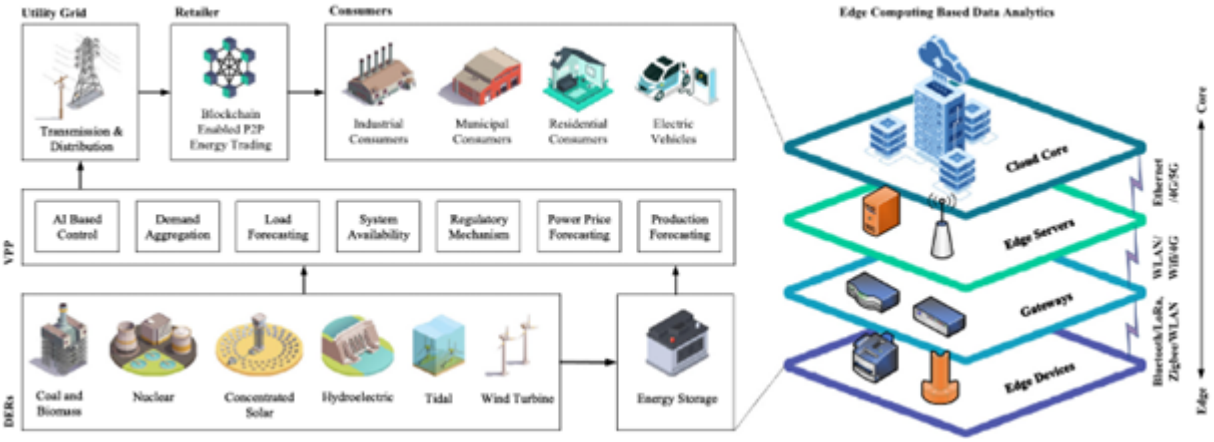
Funkce mohou být řízené operátorem, mezi které patří základní funkce typu připojení a odpojení od sítě nebo sběr dat. Kromě těchto má operátor možnost manuálně nastavit výkon, účiník nebo rychlost nabíjení a vybíjení připojené baterie. Důležitým prvkem SI jsou funkce autonomní, které střídač vykonává samostatně na základě naměřených dat. Může se jednat například o dynamickou regulaci výkonu, aby byly zachovány požadavky stanovené operátorem. Poslední druhem jsou funkce řízené nezávislou proměnnou. Principiálně jsou shodné s autonomními, akorát vstupní data, na která reagují, nezískává přímo z elektrického spojení, ale jsou dodávána přes nějaké komunikační rozhraní (např. cena, teplota, informace o zátěži ad.). Z hlediska svého účelu se dělí na celkem pět kategorií. Základní funkcí je monitorování a plánování, zbylé čtyři napomáhají řízení frek-

vence, činného výkonu, jalového výkonu a napětí v síti. [23][24]

2.2 Počítačové technologie

Počítačová síť je klíčovou součástí VPP, ale zároveň se jedná o jednu z nejproblematičtějších. Mezi typické problémy patří bezpečnost, nedostatečná kapacita, přerušení spojení, obtížná konfigurace atd. Jednou z moderních způsobů realizace komunikační sítě je blockchain technologie, která umožňuje decentralizované skladování informací, šifrování nebo peer-to-peer přenos neboli komunikaci přímo mezi klienty. Implementace této technologie je ale stále složitá vzhledem k obtížné škálovatelnosti, rychlosti a nákladům na regulaci.

V případě VPP se převážně uvažuje o počítačových sítích založených na cloud computingu, tudíž zpracování a ukládání dat a komunikace probíhá přes servery, které jsou součástí VPP. Cloudové počítače tedy fungují jako zákazník flexibilní zákazník VPP, který může v případě potřeby autonomně řídit svou spotřebu. Nevýhodou této technologie je složitý chod sítě vzhledem k obrovskému množství dat a samozřejmě i bezpečnostní nedostatky. Možným řešením je implementace edge computingu, která umožní data zpracovat alespoň částečně hned u zdroje, přímou komunikaci mezi prvky sítě a krátkodobé ukládání dat. Implementace této technologie je ovšem podstatně složitější a nákladnější. Předpokládaná infrastruktura VPP využívající edge computing je na obr. 2.3.



Obrázek 2.3: Předpokládaná infrastruktura budoucích VPP [19]

2.3 Peer-to-peer obchod s energií

Peer-to-peer (P2P) znamená obchod s energií přímo mezi zákazníky. Z hlediska VPP se o tomto způsobu obchodování s energií uvažuje v případě tzv. federativních elektráren (Federated Power Plant, FPP), což je vlastně virtuální elektrárna, která vzniklá koordinací

několika majitelů decentralizovaných zdrojů (dále jen prosumers, jakožto spotřebitelé a zároveň výrobci energie).

Důležitou podmínkou funkce tohoto konceptu je vhodná koordinace decentralizovaných zdrojů, pro kterou existují dva typy strategií: přímé a nepřímé. U přímé strategie jsou zdroje pod kontrolou operátora, který řídí jejich chod podle požadavků majitele a současných možností zdroje. Díky tomu jsou rychle a přesně řízeny parametry výroby, ale v případě P2P nás tato koncepce nezajímá, pokud se nejedná o určitou kombinaci s nepřímou strategií.

V případě nepřímé strategie jsou prosumerovi posílány pobídky, na jejichž základě provede rozhodnutí o provozu svého zdroje. Každý zdroj je tedy řízen pouze podle uvážení prosumera nezávisle na operátorovi a z hlediska komunikace je dostačující pouze jednosměrný přenos signálu. Naprosto základním příkladem tohoto principu je tzv. „time-of-use pricing“ (TOU), při kterém se cena energie mění podle spotřeby. Čím vyšší je celková spotřeba, tím vyšší je i cena energie. To pobízí zákazníka k přesouvání provozu svých zařízení do mimošpičkového období a zrovnoměrnění denního diagramu spotřeby. Skutečná aplikace samozřejmě není tak jednoduchá, jelikož stejné pobídky mohou vést k přesunu zátěže do stejného období a vzniku nové špičky. Proto se uvažují pokročilejší metody s plánováním ceny den předem, rozdílnou cenou v závislosti na lokaci, nelineární závislostí ceny na spotřebě nebo úplně náhodné nabídky cen pro různé skupiny prosumerů. Řízení zdrojů v závislosti na lokaci je zejména důležité pro redukci ztrát, zlepšení regulace napětí a dodržení provozních parametrů. Nepřímá strategie koordinace decentralizovaných zdrojů je tedy velmi důležitá pro zajištění optimálního provozu sítě v případě P2P obchodování. Bez ní by každý prosumer provozoval své zdroje a zátěže libovolně, což by vedlo k zvýšení nestability sítě.

P2P obchod by měl být oboustranně výhodnou záležitostí. Komunity prosumerů budou stejně jako provozovatel distribuční sítě usilovat o vysokou kvalitu a spolehlivost dodávky energie, jelikož optimální provoz redukuje ztráty a zvyšuje zisk. Prosumeři navíc získají úplnou kontrolu nad volbou zdroje, který pro ně energii produkuje. Tento systém by měl tedy být zejména pro společnosti podstatně atraktivnější oproti současnému obchodu se zelenými certifikáty, které pouze zaručují, že dodávaná elektřina byla vyrobena z OZE. Mimo ekologické výhody mohou P2P platformy umožnit i rozvoj filantropie v energetice, jelikož prosuměři budou schopni energii darovat.

2.3.1 Federativní elektrárny

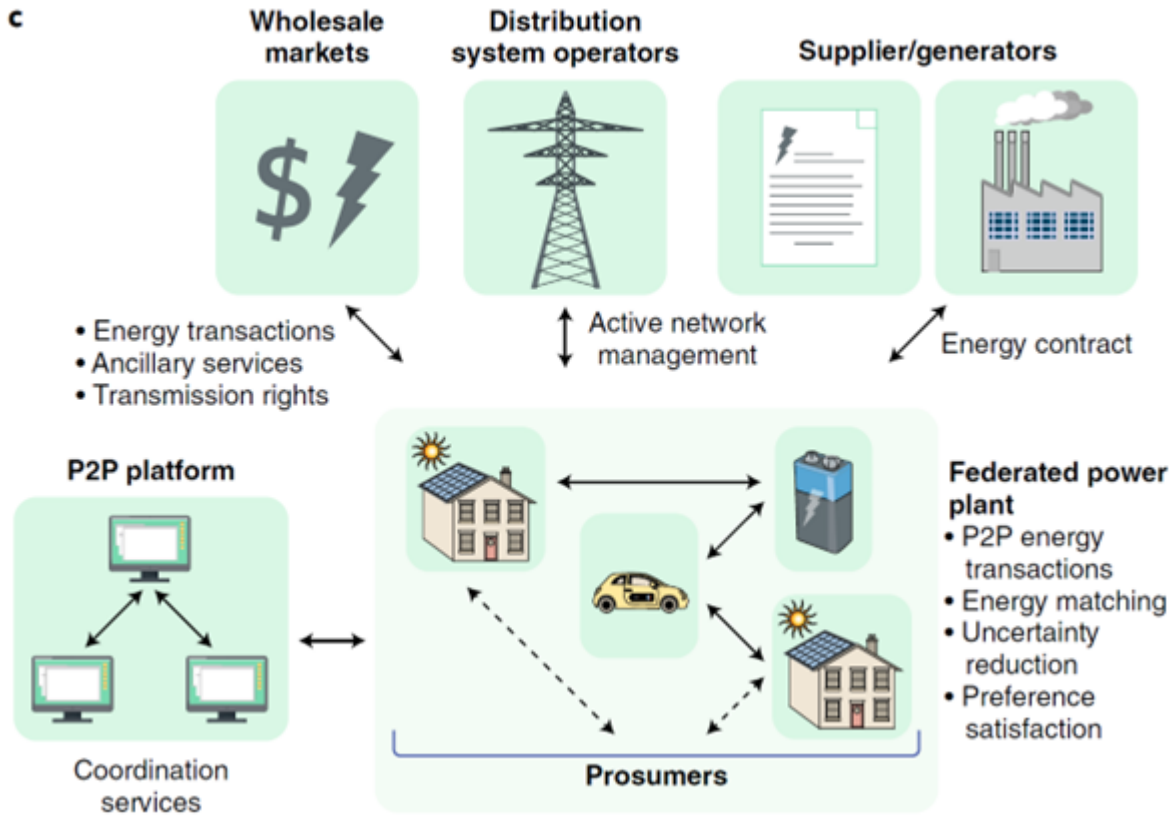
Jedním z perspektivních aplikací P2P platforem je zprostředkování transakcí se službami v síti mezi dodavateli, distributory a dalšími subjekty na jedné straně a skupinou prosumerů (FPP) na straně druhé. P2P platforma sleduje obchodování mezi prosumery, analyzuje jejich preference a možnosti a na jejich základě identifikuje, které podpůrné služby by prosumeři mohli zprostředkovat. Následně vyhodnotí cenu a vygeneruje kontrakty o provedení těchto služeb. Tyto kontrakty mohou být potenciálně dlouhodobé a zajistit skupině prosumerů výhodnější podmínky, než kdyby je každý vyjednával samostatně. Struktura kontraktů pro optimální incentivu prosumerů je zatím oblastí výzkumu. Uvažovanou možností jsou kontrakty, které specifikují množství energie, které je třeba dodat v daném časovém intervalu a případně další požadavky (např. lokace). Prosumeři by pak s těmito kontrakty obchodovali.

Princip FPP je znázorněn pomocí diagramu na obr. 2.4. Prosumeři spolu komunikují a vyměňují energii, čímž se lokálně vyrovnává výroba a spotřeba, redukuje se nejistota generace a uspokojují se preference účastníků. Obchod s energii je zprostředkován P2P platformou. Za základě obousměrného toku informací mezi dodavateli a FPP jsou tvořeny kontrakty nabízené prosumerům. S kontrakty a energií se obchoduje na trhu. Operátor využívá informace z FPP a komunikaci s prosumery pro aktivní řízení distribuční sítě.

Technologie P2P obchodu s elektrickou energií sice nabízí mnoho výhod, ale pro její rozvoj se stále nutné vyřešit potenciální problémy. Zatím není jasné, jak moc bude tato technologie ekonomicky výhodná, jak bude zajištěno, že výměna transakce s energií nepřekročí fyzické meze lokální distribuční soustavy, jakým způsobem nejlépe motivovat prosumery, aby se zapojili nebo jestli budou prosumeři schopni tuto technologii efektivně využívat. Vzhledem k těmto otázkám je nutný další výzkum hlavně v oblasti identifikace vhodných skupin zdrojů pro zprostředkování podpůrných služeb, vytvoření optimálního trhu, který bude uvažovat s různými charakteristikami decentralizovaných zdrojů (např. flexibilita, proměnlivost, výkon, lokace) a taktéž úprava legislativy, aby bylo vůbec možné tyto platformy provozovat. [25]

2.3.2 Pilotní P2P projekty

V oblasti P2P již vzniklo několik pilotních projektů s čtyřmi modely platforem obchodu s energií. První možností je nabídka služby P2P přímo od dodavatele, čímž umožňuje prosumerům vytěžit více ze svého zdroje. Druhým typem jsou platformy od obchodníků s decentralizovanými zdroji, kteří si tím mohou zvýšit hodnotu svého produktu. Třetí možností jsou komunitní platformy. Ty nalézají vyžití například v mikrosítích, kde mo-



Obrázek 2.4: Diagram kooperace mezi členy FPP a s nadřazenými subjekty [25]

hou pomoci udržet stabilní dodávku energie po odpojení od distribuční sítě. Poslední možností, která může být spojená s předcházejícími typy jsou blockchainové platformy. Ty slouží bezpečnému managementu a provádění transakcí pomocí chytrých kontraktů mezi prosumery bez nutnosti ověřené třetí strany. Vybrané pilotní projekty reprezentující každý typ platformy jsou následující:

- Piclo

Projekt ve Spojeném království, kde si zákazníci mohou koupit elektřinu přímo od lokálních obnovitelných zdrojů. Zákazník si vybere, které zdroje upřednostňuje a na základě této preference, výrobních možností a dalších informací je optimálně nastavena produkce.

- SonnenCommunity

Projekt zavedený německým výrobcem baterií sonnenBaterie. Jedná se o komunitu vlastníků těchto baterií, kteří mezi sebou mohou sdílet vyrobenou energii. Přebytky výroby jsou ukládány do virtuální baterie, ze které může každý člen odebírat v případě nedostatku vlastní výroby. Vše je řízeno centrálním softwarem.

- TransActive Grid

Projekt v Brooklynu umožňující členům mikrosítě nakupovat a prodávat energii pomocí chytrých kontraktů a blockchain technologie.

- Electron

Platforma pro obchodování s energií pomocí blockchain technologie.
[26]

2.4 VPP projekty

VPP není novým konceptem, poprvé přišel s nápadem virtuální elektrárny Shimon Awerbuch již v roce 1997. [27] V dnešní době však VPP nabývají čím dál více na důležitosti. V roce 2019 bylo odhadnuto, že trh s VPP bude v letech 2020 až 2027 zvyšovat svou hodnotu o 21,3 % ročně. [28] Nejvýznamnější projekty virtuálních elektráren jsou shrnuty v tabulce 2.1.

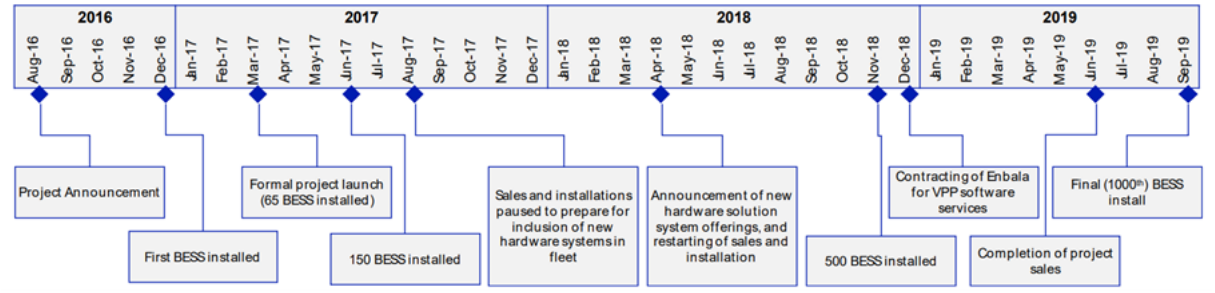
Tabulka 2.1: Seznam nejvýznamnějších projektů VPP [19][29]

Název	Začátek	Země	Typy DERs	Počet DERs	Počet zákazníků	Výkon
FENIX	2005	Spojené království, Španělsko, Francie, ad.	Malé kogenerační jednotky, FV, větrné el.	10000x	169000	1234 MVA
EDISON	2011	Dánsko	Elektromobily	52	27000	125 MW
WEB2ENERGY	2009	Německo, Polsko, ad.	Kogenerace, FV, Větrné el., bioplyn, vodní el.	16	200	40,5 MW
SA VPP	2017	Austrálie	FV, baterie	1000	50000	250 MW
CON EDISON VPP	2016	USA	FV, baterie	1000	300	100-300 MW

2.4.1 SA VPP

SA VPP neboli virtuální elektrárna v jižní Austrálii je projekt oznámený v roce 2016. Cílem bylo instalovat 1000 bateriových systémů v Adelaide. Původně byl hlavním technologickým partnerem Sunverge, později se přidali další partneři LG Chem, Solar Edge a Tesla. Pro vytvoření softwaru na řízení celé VPP byla kontraktována společnost Enbala. Řízení každé jednotky má být na bázi „cloud to cloud“ integrace neboli jsou řízeny přes propojené cloudy, které automaticky sdílí informace. Časová osa projektu až do září 2019 je na obr. 2.5. Po dokončení instalace 1000 baterií oznámila Tesla třetí fázi projektu, ve které plánuje nainstalovat dalších 3000 bateriových systémů. Do budoucna se plánuje 50000 instalací. [30]

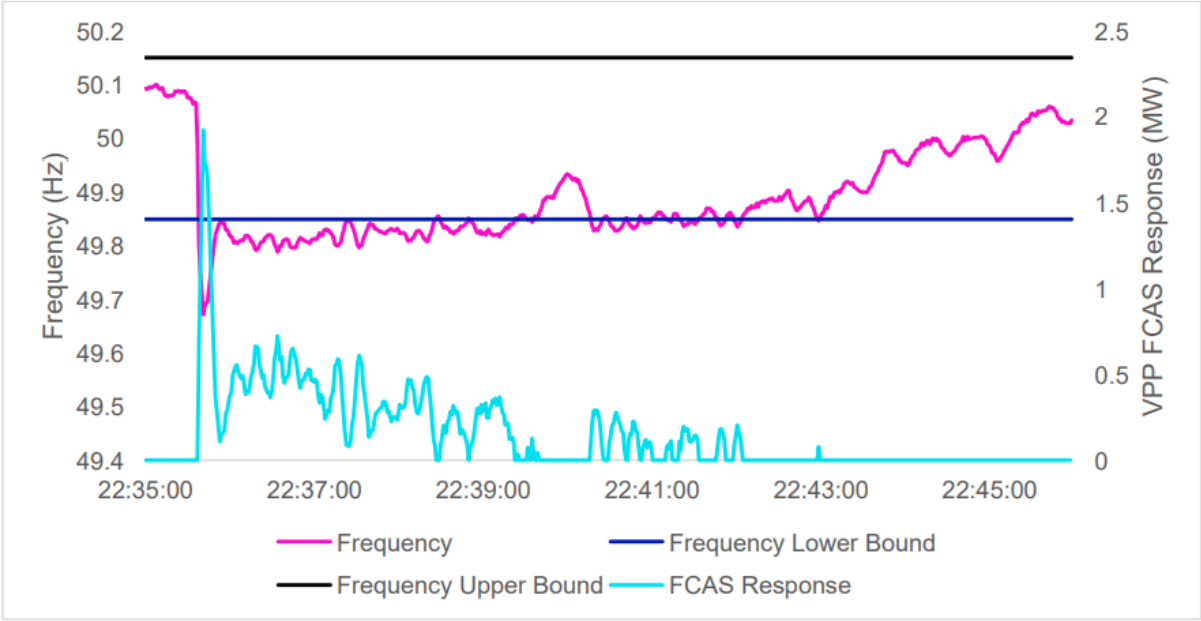
Realizace projektů nebyla jednoduchá a již v raných fázích bylo třeba překonat mnoho výzev. Jedním z prvních problému bylo nalezení vhodných zákazníků, aby instalace baterie na daném místě smysluplně přispěla virtuální elektrárně a zároveň, aby se zákazníkovi investice vrátila. Mezi podmínky tedy patřilo např. dostatečné množství přebytků, vhodná



Obrázek 2.5: Časová osa projektu VPP-SA [31]

lokalita, žádný výkupní tarif atd. Další problémy nastaly v oblasti instalace hardwaru, zejména měřících zařízení, jelikož se jednalo o nové, málo prověřené technologie. Problémy, které přetrvávají, je odchod zákazníků od projektu převážně kvůli změně bydliště nebo negativní vliv změn norem, legislativy a smluvních podmínek provozovatelů soustavy. Příkladem je normativní stanovení maximálního celkového výkonu střídačů na 10 kVA v roce 2017, což by zabránilo některým zákazníkům v instalaci nových bateriových systémů. Norma nakonec byla upravena, ale byla ukázána důležitost spolupráce mezi všemi subjekty v oblasti energetiky z hlediska podpory progresivních projektů.

V průběhu projektu bylo otestováno mnoho služeb očekávaných od VPP. Začalo se nejjednoduššími typu centrálně řízeného plošného nabíjení a vybíjení bateriových systémů, postupně byla vyzkoušena i tvorba lokálních režimů, které umožňují bateriovým systémům autonomně reagovat na změny a nestability v síti. Jedním z podstatných výsledků projektu je demonstrace schopnosti poskytnout podpůrné služby, a to například pro řízení frekvence (Frequency Control Ancillary Services, FCAS) při nepředvídatelných odchylkách (při výpadku velké elektrárny či závažnější poruše v síti). Tesla baterie tuto službu poskytují lineárním navýšením nebo snížením výkonu s různým sklonem podle velikosti odchylky. Situace, kdy bylo využití služby FCAS potřeba, je ukázána na obr. 15. Síťová frekvence kvůli výpadku náhle klesla pod 49,7 Hz. VPP proto poskytlo dodatečných 1,9 MW výkonu, čímž pomohla vrátit velikost frekvence do požadovaných mezí. Další výsledky projektu nejen z hlediska technologického, ale i ekonomického nebo marketingového lze nalézt ve zprávě [31].



Obrázek 2.6: Ukázka regulace odchylky frekvence SA VPP [31]

Kapitola 3

Návrh umělé neuronové sítě pro statické prostředí

Pro návrh umělé neuronové sítě jsem zvolil prostředí Matlab, jelikož nabízí mnoho nástrojů strojového učení v oblasti neuronových sítí. Hlavními z těchto nástrojů je toolbox hlubokého učení, který obsahuje různé aplikace pro tvorbu mělkých a hlubokých umělých neuronových sítí a taktéž reinforcement learning toolbox pro tvorbu řídících procesů a rozhodovacích algoritmů. Základní čtyři nástroje toolboxu pro hluboké učení je analýza a predikce časových řad, shlukování dat, rozpoznávání vzorů a fitování dat.

Nástroj shlukování dat využívá neuronovou sít pro rozdělení dat do specifických podskupin podle jejich společných charakteristik. Může se proto využívat například k třídění fotografií. Také je možné ho kombinovat s nástrojem pro rozpoznávání vzorů, který je právě schopen hledané charakteristiky detekovat.

Pro predikci časových řad nabízí Matlab možnost řešení tří problémů. Nejjednodušším problémem je predikce budoucích hodnot časové řady $y(t)$ podle hodnot předcházejících nazývaná NAR. Zkratka NAR vychází z anglického nonlinear autoregressive neboli nelineární autoregresní. Funkce sítě NAR se zapisuje následovně:

$$y(t) = f\left(y\left(t-1\right), ..., y\left(t-d\right)\right). \tag{3.1}$$

Druhým problémem je predikce budoucích hodnot časové řady $y(t)$ na základě předcházejících hodnot této a sekundární řady $x(t)$. Tento problém se nazývá nelineární autoregresní s externím vstupem (zkratka NARX z anglického nonlinear autoregressive with external input). Sítě řešící tuto predikci tedy analyzují časové řady, jejichž hodnoty jsou ovlivňovány dalšími časově proměnnými parametry. Jejich funkce se zapisuje takto:

$$y(t) = f\left(y\left(t-1\right), ..., y\left(t-d\right), x\left(t-1\right), ..., x\left(t-d\right)\right). \tag{3.2}$$

Poslední typ predikce je podobný NARX avšak tentokrát známe pouze předcházející hodnoty $x(t)$. Tento typ nemá speciální název. Označuje se jako nelineární vstup-výstup a pro jeho funkci platí:

$$y(t) = f\left(x\left(t-1\right), ..., x\left(t-d\right)\right). \tag{3.3}$$

3.1 Nástroj pro fitování dat

Jedním z nejuniverzálnějších nástrojů je fitovaní dat neboli nalezení vztahu mezi libovolnou sadou vstupních a výstupních hodnot. Aplikace toolboxu hlubokého učení k tomuto využívá mělkou dvouvrstvou dopřednou neuronovou síť s možností výběru ze tří trénovacích algoritmů. Základním algoritmem je Levenberg-Marquart, který je sice nejrychlejší, ale zároveň vyžaduje nejvíce paměti. Tento algoritmus počítá následující krok úpravy parametrů sítě podle vztahu:

$$x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T e, \tag{3.4}$$

kde J je Jakobián neboli matice obsahující parciální derivace odchylek sítě vůči vahám a biasům, e je vektor odchylek, I je jednotková matice a μ je skalární parametr. Součin $J^T e$ je roven gradientu g. Pokud je μ nulový, jedná se o Newtonovu metodu, která je rychlejší a přesnější. Cílem tohoto algoritmu je tedy co nejrychleji snížit hodnotu μ.

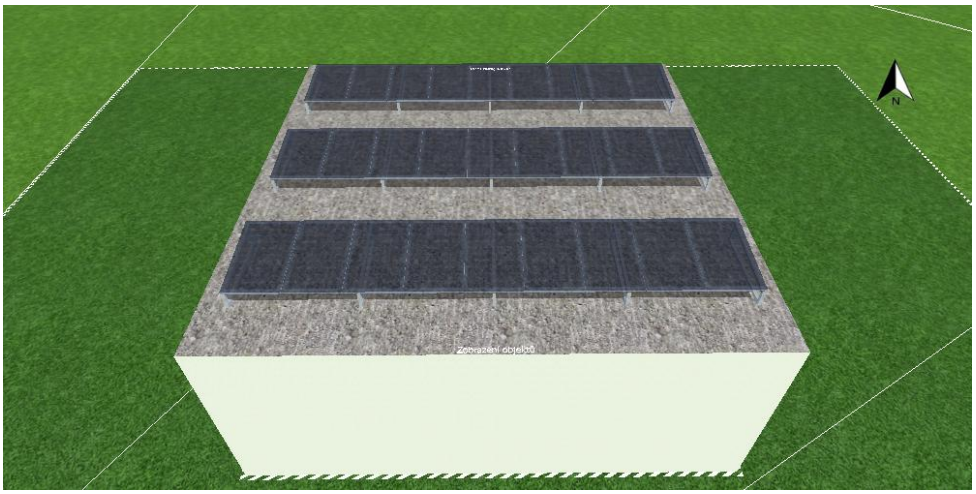
Druhou možností je Bayesovská regularizace (Bayesian regularization), která využívá stejně jako předcházející Levenberg-Marquardtovy optimalizace k úpravě vah a biasů. Následně ale navíc minimalizuje kombinaci kvadratických odchylek a vah pro nalezení správné kombinace pro lepší generalizaci neuronové sítě. Algoritmus je sice pomalejší, ale vhodnější právě k dobré generalizaci malých datových množin. Posledním algoritmem je SCG (Scaled conjugate gradient backpropagation), který se využívá pro větší datové množiny, jelikož gradientní výpočty umí efektivněji pracovat s pamětí oproti Jakobiánským výpočtům předchozích dvou algoritmů. [10] [32][33]

3.2 Testovací model fotovoltaického systému

Pro návrh fotovoltaického systému byl zvolen software PV*Sol. Software umožňuje namodelovat různé fotovoltaické systémy. Z typů systému nabízí výběr ostrovního nebo

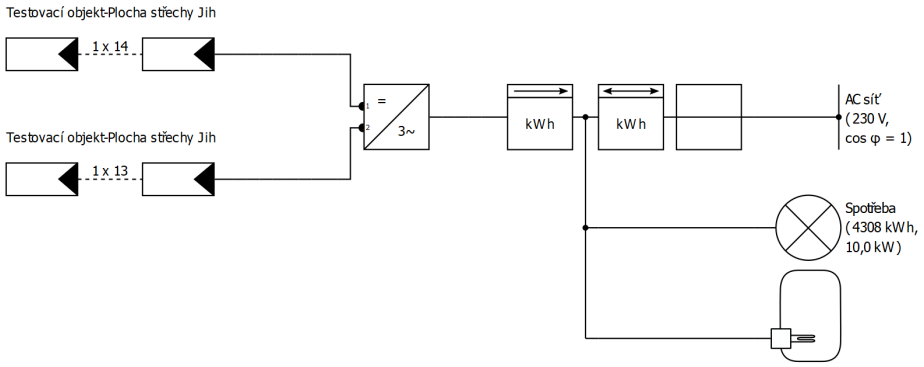
připojeného k síti, podle způsobu akumulace přebytků (topné těleso, baterie, elektromobil) a jejich různé kombinace. Po zvolení typu systému a nastavení základních parametrů simulace se volí spotřební profil, kde si lze vybrat buďto jeden z výchozích nebo importovat vlastní data. Poté se zvolí parametry akumulace (například rozměry a teploty akumulační nádrže). Pro modelování fotovoltaického pole a jeho umístění je podprogram 3D vizualizace, kde je možné navíc přidat okolní objekty pro analýzu zastínění a nakonfigurovat zapojení. K dispozici je rozsáhlá databáze fotovoltaických modulů a střídačů od různých výrobců. Na základě navrženého modelu a dalších nastavení PV*Sol vygeneruje výkresy. Závěrem je nastavení různých ekonomických parametrů typu investice, úroky, odkup elektřiny atd.

Pro počáteční testování byl navržen jednoduchý systém s fotovoltaickými moduly Cheetah Perc JKM325M-60-V. Moduly jsou složeny z monokrystalických článků a dosahují jmenovitého výkonu 325 W. Modelový objekt je budova s plochou střechou. Celkem je na této střeše umístěno 27 modulů o celkovém špičkovém výkonu 8,78 kWp. Panely jsou rozmístěny na kovových konstrukcích ve třech řadách po devíti panelech dostatečně vzdálené, aby na sebe vzájemně nestínily (viz obr 3.1). Celé pole je orientováno přímo na jih se sklonem 45 ° v úvodním nastavení. Pro transformaci napětí je použit jeden centrální střídač GoodWe GW8000-DT o celkovém výkonu 8 kW. Střídač je vybaven dvěma trackery bodu maximálního výkonu, jeden je napojen na prvních 14, druhý na zbylých 13 modulů.



Obrázek 3.1: Situace modelového fotovoltaického systému se sklonem 0 °

Pro testovací objekt byl zvolen výchozí profil spotřeby 2 osob se 2 dětmi o celkové roční spotřebě 4308 kWh se špičkovým zatížením 10 kW. Přebytky systému jsou ukládány do ohřevem vody v akumulační nádrži s kapacitou 8372 kWh a topným tělesem o výkonu 2 kW. Celkové schéma systému je znázorněné na obr. 3.2).



Obrázek 3.2: Schéma modelového fotovoltaického systému.

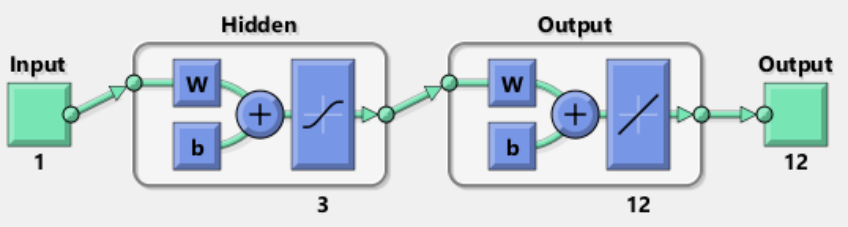
Pro simulaci bylo vybrány data Meteonorm 8.1 z let 1996-2015 v Praze. Průměrná roční teplota je v tomto případě 9,2 ° C. Albedo uvažujeme 20 %, roční suma globálního záření je 1055 kWh/ m^2, znečištění panelů není uvažováno. Simulace může být provedena buďto v hodinových nebo minutových krocích. Hodinová simulace postačí pro návrh prvotní mělké neuronové sítě, kdy nás bude zajímat pouze agregovaná produkce za delší časové období. V případě řízení přebytků či jiných aplikací, které analyzují systém v průběhu dne je již potřeba minutové rozlišení dat.

Výstupem simulace jsou časové průběhy mnoha veličin. Software uvažuje ztráty stítěním, odchylkou od jmenovité teploty, diod, kabeláže, měniče ad. Z hlediska ohřevu vody v bojleru odhaduje i tepelné ztráty na základě zadaných parametrů. U parametrů bojlerů se specifikují rozsahy teplot, rozměry, tloušťka izolace a maximální výkon topného tělesa. Spotřebu teplé vody lze nastavit pouze celkovou roční a vybrat k ní jeden z výchozích profilů spotřeby. Vlastní profil spotřeby tedy importovat nelze. Software PV*Sol tedy k analýze využití přebytků příliš vhodný není.

3.3 Návrh zkušební mělké neuronové sítě

Software PV*Sol má spoustu nedostatků. Sice je schopen navrhnout optimální konfiguraci střídače s MPP trackery, ale optimalizace z hlediska základních parametrů fotovoltaického pole jako je sklon panelů, azimut, rozmístění panelů je velmi obtížná a zdlouhavá. Samotný software totiž tuto optimalizaci nenabízí a je třeba manuálního hledání. Tomu nenapomáhá ani fakt, že každá změna jakéhokoli parametru montáže modulů se provádí v podprogramu 3D návrhu a po jejím provedení se musí opět nastavit konfigurace střídačů a přepočítat celá simulace, než získáme přístup k výsledný datům produkce. Tento proces je tedy extrémně zdlouhavý a urychlení tohoto procesu se nabízí jako vhodná ukázka schopností umělých neuronových sítí.

Pro ukázkovou umělou neuronovou síť na optimalizaci fotovoltaického systému jsem za vstupní parametr zvolil sklon modulů a za výstupní parametr intenzitu záření dopadající na plochu modulu v průběhu roku v každém měsíci. Vstupem sítě je tedy vždy jedna hodnota, zatímco výstupem je 12 hodnot. Celkem jsem vygeneroval data pro sklony od 0 do 90 ° s krokem 10 °, tedy 10 vstupních hodnot. Celková datový soubor tedy obsahuje 10 vstupů a 120 výstupů. Pro nalezení vztahu mezi vstupem a výstupem jsem použil dvouvrstvou mělkou neuronovou síť, jejíž schéma je na obr. 3.3.



Obrázek 3.3: Schéma použit0 umělé neuronové sítě v prostředí Matlab

Použitá data je vhodné nejprve normalizovat a případně i logaritmovat. V tomto případě jsem tento krok vynechal, jelikož se nejedná o velké množství dat ani extrémní hodnoty, a normalizace tedy nemá příliš velký význam. Následně je třeba data rozdělit do tří podskupin: trénovací, validační a testovací. Na trénovacích datech se síť naučí hledaný vztah a na validační skupině kontroluje svou odchylku. Testovací soubor slouží ke kontrole generalizace sítě. V tomto případě bylo 60 % dat trénovacích, 30 % validačních a 10 % testovacích.

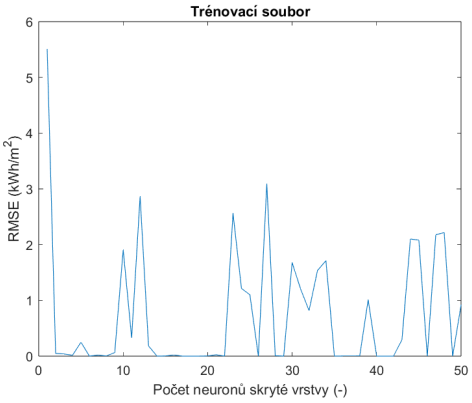
Pro získání nejlepších výsledků neuronové sítě, je vhodné zjistit optimální počet neuronů ve skryté vrstvě. Toho lze dosáhnout jednoduchým for cyklem, při kterém se pro každou síť vypočítá směrodatná odchylka (Root-mean-square error, RMSE) podle vztahu:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N} (y_n - y_{tn})^2}{N}}, \tag{3.5}$$

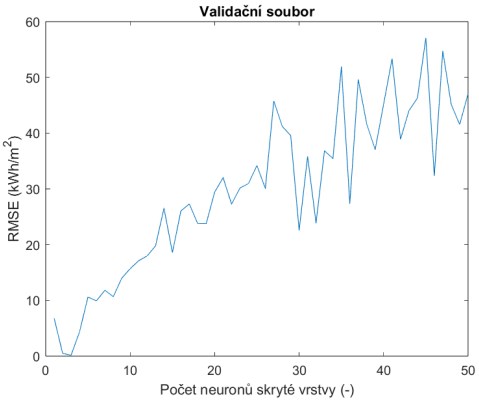
kde y_n je n-tá výstupní hodnota vypočtená neuronovou sítí a y_{tn} je n-tá skutečná výstupní hodnota z dat simulace. Výsledný kód v matlabu poté vypadá následovně:

```
%Trenovani siti s ruznymi pocty neuronu skryte vrstvy
xt=xr';
yt=yr';
for i = 1:50
    hlayersize = i;
    net = fitnet(hlayersize);
    net.divideParam.trainRatio=60/100;
    net.divideParam.valRatio=30/100;
    net.divideParam.testRatio=10/100;
    [net,tr]= train(net,xt,yt);

    yTrain = net(xt(:,tr.trainInd));
    yTrainTrue = yt(:,tr.trainInd);
    yVal = net(xt(:,tr.valInd));
    yValTrue = yt(:,tr.valInd);
    yTest = net(xt(:,tr.testInd));
    yTestTrue = yt(:,tr.testInd);
    rmse_train(i) = sqrt(mean((yTrain-yTrainTrue).^2,"all"));
    rmse_val(i) = sqrt(mean((yVal-yValTrue).^2,"all"));
    rmse_test(i) = sqrt(mean((yTest-yTestTrue).^2,"all"));
end

%Nalezeni optimalniho poctu neuronu skryte site
[M1,I1] = min(rmse_train)
[M2,I2] = min(rmse_val)
[M3,I3] = min(rmse_test)
[M,I] = min(rmse_train+rmse_val+rmse_test)
```

Hodnoty RMSE pro různé počty neuronů skryté vrstvy v případě trénovacího a validačního souboru dat jsou na obr. 3.4. Z grafu je patrné, že přestože u trénovacího souboru dosahují i větší skryté vrstvy nízké chyby, minimální chyba u validačního souboru je pouze pro malé množství neuronů a s rostoucím množstvím rychle roste. Při vyšším počtu totiž fituje neuronová síť trénovací data příliš přesně a nedosahuje dostatečné generalizace. Minimální suma RMSE pro všechny datové soubory vyšla 0,3940 kWh/m^2 pro skrytou vrstvu o třech neuronech. Nejmenší RMSE $1{,}9 \cdot 10^{-14}$ kWh/m^2 dosáhl podle očekáváná soubor trénovacích dat, podstatně vyšší chybu 0,16 kWh/m^2 měl soubor validačních dat a nejvíce se opět odchyloval testovací soubor dat s RMSE = 0,2 kWh/m^2. To znamená,



(a) RMSE pro trénovací soubor dat



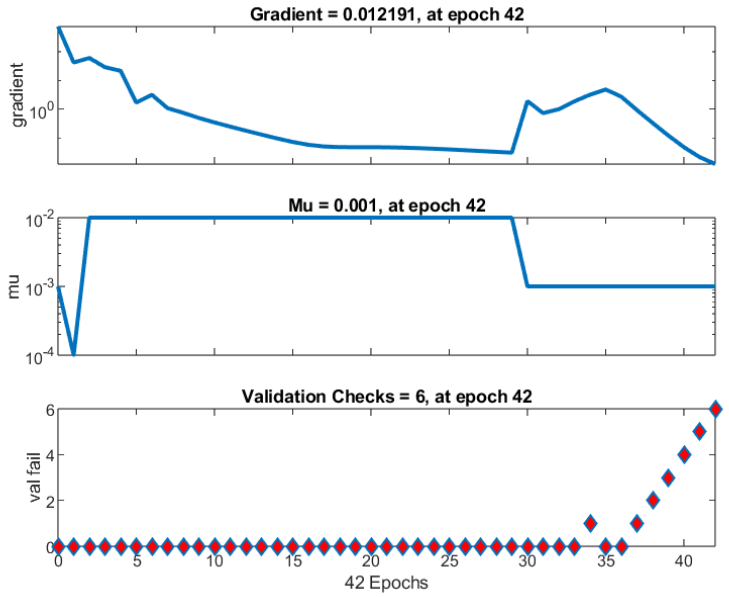
(b) RMSE pro validační soubor dat

Obrázek 3.4: RMSE v závislosti na velikosti skryté vrstvy

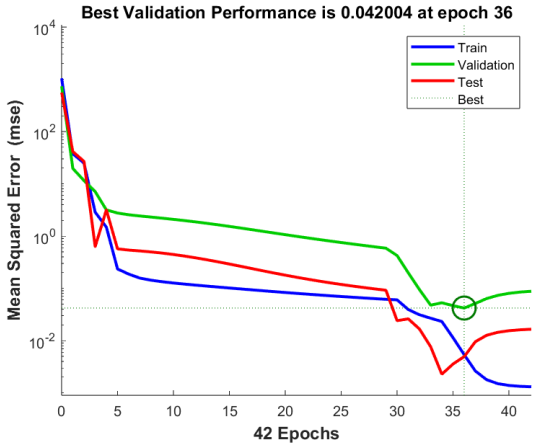
že v nejhorším případě se výsledná hodnota neuronové sítě lišila pouze řádově o desetiny procenta oproti hodnotám skutečným. Je nutné však dodat, že opakované trénování neuronové sítě nedospěje téměř nikdy ke stejným výsledkům. Proto tato výsledná hodnota celkové chyby platí pouze pro jeden z mnoha procesů učení, které jsem se sítí zkoušel. Řádově však RMSE vždy vycházelo stejně.

Neuronová síť je učena pomocí Levenberg-Marquardtova algoritmu, pro který je nastaveno několik ukončovacích podmínek. Buďto je překročen maximální počet epoch (počet opakování výpočtu algoritmu) nebo maximální stanovený čas. V takovém případě pravděpodobně nebude dosaženo uspokojivých výsledků. Totéž platí v případě, že parametr μ překročí svou maximální hodnotu. Pro získání požadovaných výsledků s dostatečnou přesností je třeba, aby byl proces učení ukončen buďto minimalizací výkonnosti nebo gradientu sítě na zadanou hodnotu nebo v případě, že odchylka validačního souboru dat se zvýší v předem specifikovaném počtu po sobě následujících epoch. V případě této neuronové sítě došlo právě k navýšení odchylky validačního souboru v šesti epochách po sobě, což je znázorněno na spodním grafu na obr. 3.5. Obrázek ukazuje průběh právě zmíněných veličin, jejichž požadované hodnoty ukončují učení. Lze si všimnout, že parametry spolu nemusejí vůbec souviset. Chyba validační souboru dat stoupala, zatímco gradient klesal a parametr μ byl konstantní. Průběh středních kvadratických chyb (Mean-square error, MSE) jednotlivých souborů naleznete na obr. 3.6. Zatímco chyba trénovacího souboru s rostoucím počtem epoch klesá, chyba validačního souboru dosáhne minima v epoše 36 a poté stoupá. Chyba testovacího souboru začne stoupat ještě rychleji.

Srovnání výstupů neuronové sítě se simulovanými výsledky pro jednotlivé soubory dat je znázorněno v příloze na obr. B.1. Křivky mají velmi blízko k závislosti $x = y$. Pro potvrzení správnosti výsledku jsem ještě manuálně do Matlabu vložil hodnoty intenzity záření, které nebyly součástí dat, se kterými pracovala neuronová síť. Jednalo se o výsledky simulací se sklonem od 5 ° do 85 ° s krokem 5 °. Směrodatná odchylka tohoto



Obrázek 3.5: Velikost parametrů pro ukončení učení neuronové sítě



Obrázek 3.6: Velikost MSE v průběhu učení neuronové sítě

souboru dat byla 0,43 kWh/m^2, tedy lehce vyšší, ale řádově stále stejná jako v případě datových souborů, na kterých se neuronová síť učila. Srovnání dvou vybraných souborů těchto dat s výstupem neuronové sítě pro dané hodnoty sklonu je v příloze na obr. B.2. Tímto jsem si ověřil přesnost sítě a mohl vypočíst optimální sklon modulů, při kterém je celková intenzita záření dopadající na plochu za jeden rok maximální. Tento sklon vyšel 34 °.

Samozřejmě i mělká neuronová síť dokáže pracovat s větším množstvím vstupů a je tedy možné tuto síť využít i k mnohem komplexnější optimalizaci mnoha parametrů najednou. Otázkou je v tomto případě rychlost učení a nároky na hardware. Pro tuto jednoduchou aplikaci probíhalo každé učení neuronové sítě pouze několik vteřin.

Kapitola 4

Návrh modelu pro využití přebytků

Pro využití přebytků není zcela vhodné provádět strojové učení pomocí pevně daných statických vstupů a výstupů. Spotřeba i výroba je totiž silně proměnlivá a takto naučená umělá neuronová síť by pravděpodobně nebyla schopna dosáhnout dostatečné obecnosti. Na tuto aplikaci je proto potřeba dynamický systém. Jako podstatně vhodnější alternativa se tedy nabízí zpětnovazební učení, kdy můžeme zadat naše požadavky ve formě různých podmínek. Těchto podmínek se bude držet vybraný agent a na jejich základě zprostředkovávat řízení systému. Každá podmínka má svou specifickou hodnotu ve formě odměny, kterou agent za její splnění dostává. Tím je zaručeno, že výsledný systém bude splňovat naše nejdůležitější požadavky i za nestandardních podmínek.

Nevýhodou realizace takového systému jsou podstatně náročné požadavky na výpočetní výkon. V této práci se v praktické části zaměřuji na aplikace, které může realizovat každý bez nutnosti přístupu k vysokému množství výpočetních jednotek nebo cloudovým službám. Proto nebude výsledný navržený model nijak komplexní a bude se spíše jednat o základ, na kterém bude možné dále stavět. O možnostech rozšíření modelu budu hovořit v závěrečné části této kapitoly. V následujících sekcích bude popsán uvažovaný fotovoltaický systém a následně uvedeny možnosti realizace zpětnovazebního učení v Matlabu. Poté je již popsán samotný navržený model.

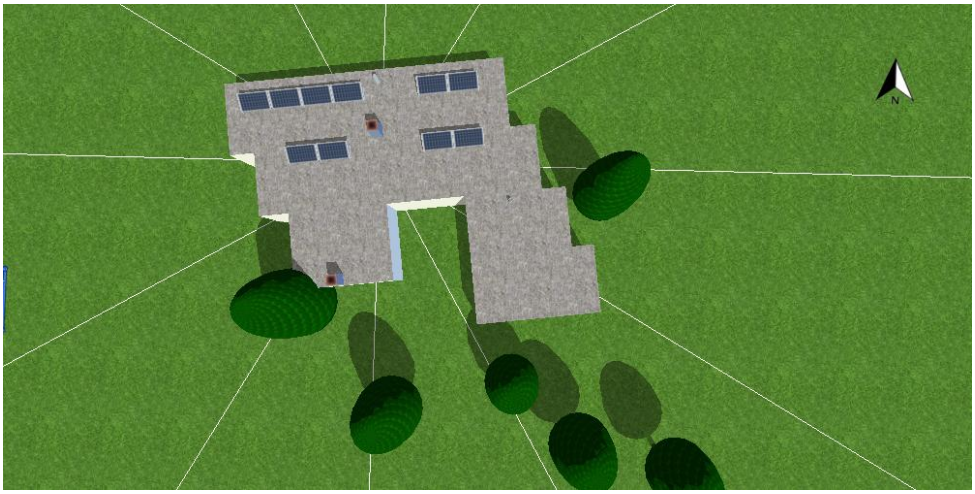
4.1 Popis zvoleného fotovoltaickém systému

Pro ověření funkce neuronové sítě na skutečném systému bylo zvolen domácí fotovoltaický systém na budově v obci Řevnice. Přesné souřadnice tohoto objektu jsou 49,9 ° N-14,2 ° E. Satelitní pohled na budovu, na kterém lze vidět i fotovoltaický systém je na obr. 4.1. Střecha budovy je téměř plochá se mírným sklonem 1,73 °, v okolí se nacházejí stromy,

které způsobují zastínění modulů.

Obrázek 4.1: Satelitní pohled na budovu s fotovoltaickým systémem

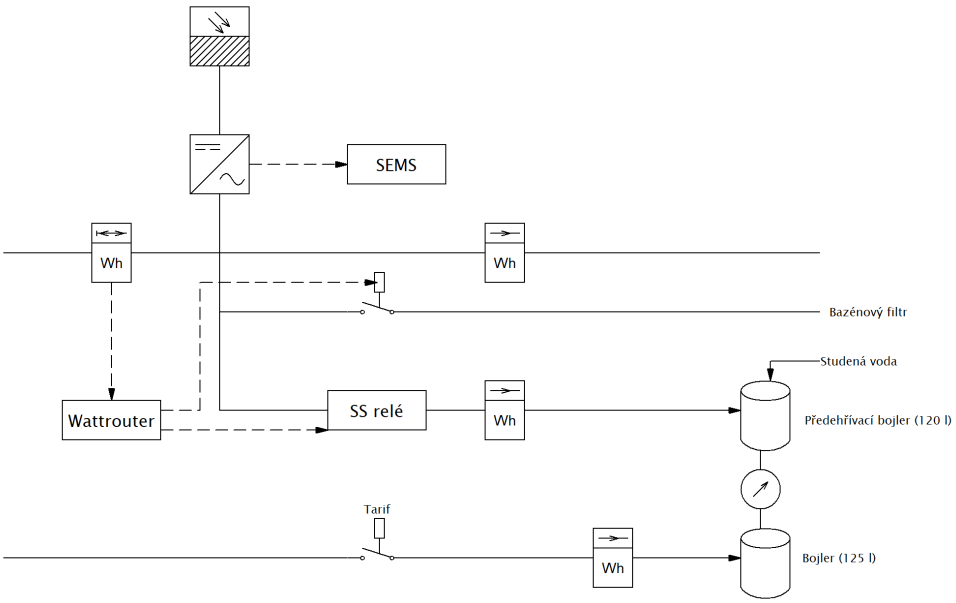
Fotovoltaické pole se skládá z 10 modulů Cheetah Perc JKM325M-60-V o jmenovitém výkonu 325 Wp. Celkový jmenovitý výkon systému je tedy 3,25 kWp. Moduly jsou umístěny na kovové konstrukci se sklonem 30 ° a azimutem 173 °. Pole je napojeno na jeden centrální střídač GoodWe 3000NS s jmenovitým výkonem 3 kW a jedním sledovačem bodu maximálního výkonu. Všechny tyto komponenty byly namodelovány v programu PV*Sol. Rozmístění panelů na střeše a okolních objektů pro simulaci zastínění lze vidět na obr. 4.2.



Obrázek 4.2: Model budovy v 3D návrhu programu PV*Sol

Přebytky fotovoltaického systému jsou využívány k ohřevu vody v předehřívacím bojleru. Ten je propojen s druhým bojlerem, který je napájen ze sítě a z něhož se již odebírá voda pro využití v domácnosti. Toky energie jsou měřeny wattmetry, jejichž zapojení je znázorněno ve schématu na obr. 4.3. První wattmetr měří spotřebu domácnosti kromě bojlerů a bazénového filtru. Druhý a třetí wattmetr měří spotřebu elektřiny příslušných bojlerů. Celková produkce elektřiny systému je měřena přes střídač a výsledná data jsou

odesílána na web SEMS Portal. Napájení předehřívacího bojleru je řízeno wattrouterem a polovodičovým relé. Mezi bojlery je umístěn měřič množství protečené vody. Spotřeba elektrické energie na všech fázích i spotřeba vody je odesílána na web Energomonitor. [34]



Obrázek 4.3: Schéma zapojení měřičů

Pro získání vstupních dat do modelu využití přebytku tedy existují dvě možnosti. Buďto data nasimulovat pomocí PV*Solu nebo vzít skutečná naměřená data. Druhá možnost se jeví jako výrazně lepší, avšak nevýhodou je nedostatečné rozlišení dat. SEMS Portal totiž slouží pouze k sledování produkce z dlouhodobého hlediska, aby měl uživatel pouze přehled o produkci a výnosu elektrárny. Proto nejmenší možné rozlišení, které nabízí, je jeden den. To je pro potřeby využití přebytků, kdy je vyžadováno udržovat teplou vodu v průběhu dne, nedostačující. Na druhou stranu se jedná o skutečná data, která žádná sebelepší simulace nenahradí. Abych se tedy co nejvíce přiblížil skutečným hodnotám a zároveň mohl data využít pro simulaci, rozhodl jsem se použít jako základ nasimulovaná data s rozlišením jedné minuty a přepočítat je úměrně skutečným naměřeným hodnotám. Takto budou alespoň odpovídat celkové produkce za jeden den u simulovaných dat hodnotám skutečně naměřeným.

V případě dat spotřeby už je rozhodování jednodušší, jelikož Energomonitor zaznamenává v minutových intervalech, tudíž dosahuje stejného rozlišení jako PV*Sol. Tady tedy můžeme použít data přímo. K dispozici jsou minutové záznamy spotřeby pro jednotlivé fáze i spotřeba vody. Pro potřeby modelu nás zajímá zejména spotřeba na první fázi neboli spotřeba budovy, abychom mohli vypočíst množství nadbytečné energie z fotovoltaického systému. Vhodné se zdají být i data o spotřebě vody, ale z důvodů výpočetní náročnosti v modelu nebudou zahrnuty fyzické přetoky vody. Podrobnější zdůvodnění na-

leznete v sekci návrhu modelu.

4.2 Vytvoření modelu zpětnovazebního učení v Matlabu

Zpětnovazební učení má v Matlabu vlastní toolbox. Tento toolbox nabízí mnoho různých algoritmů učení a druhů agentů. Obdobně jako v případě učení mělkých neuronových sítě je zde možnost provádět jednodušší návrhy pomocí aplikace a komplexnější v prostředí příkazového řádku a Simulinku.

Ať už zvolíme jakýkoli způsob realizace, pracovní postup bude vždy stejný. Nejprve se vytvoří prostředí, ve kterém bude agent existovat. Prostředí analyzuje vliv akcí agenta na dosažení žádaných výsledků a generuje pro něj odměnu. Zároveň produkuje data o svém stavu a předává je agentovi. Po úspěšném stanovení prostředí následuje definice odměn složená z naplánování požadavků na řízení a vymyšlení jejich vypočtu z veličin v prostředí. Tím získáme potřebné okolí, do kterého lze implementovat agenta. Agent je v matlabu struktura obsahující algoritmus učení a svou strategii. Strategie je vlastně zmapování vztahu mezi současným stavem a pravděpodobnostním rozložením potenciálních akcí agenta. Nejčastěji se reprezentuje pomocí hlubokých neuronových sítí nebo v případě jednodušších aplikací pomocí vyhledávácích tabulek. Algoritmus učení upravuje parametry strategie na základě akcí, stavů a odměn tak, aby vznikla optimální strategie dosahující maximální kumulativní odměny. Pro ladění strategie využívá agent tzv. funkční aproximátory. Ty lze využít dvěma způsoby. Prvním z nich je tzv. kritik, který za dané stavy a akci vrátí předpovězenou diskontovanou hodnotu kumulativní dlouhodobé odměny. Druhý je tzv. herec, který na základě stavu vyprodukuje akci, která maximalizuje předpovězenou diskontovanou odměnu. Kritici jsou lepší pro diskrétní prostředí, v kontinuálním prostředí jsou výpočetně náročnější. Naopak agenti s herci jsou jednodušší, a proto jsou vhodnější pro kontinuální prostředí. Nevýhodou je citlivost na šum a riziko konvergence v lokálních minimech. Samozřejmostí je možnost kombinace obou typů, kdy herec volí nejlepší odměnu podle zpětné vazby od kritika. Další rozdělení agentů je podle jejich přístupu ke strategii. Agent se buďto striktně drží své strategie a upravuje pouze ji nebo může vyhodnocovat a upravovat i strategii, kterou zrovna nevyužívá. Prvním typem je například již dříve zmiňované SARSA, zástupcem druhého typu je Q-learning.

Při výběru agenta je nutné přihlédnout na všechny výše zmíněné parametry. Jelikož model bude pracovat s diskrétními daty spotřeby a produkce, je určitě vhodné zvolit agenta pro diskrétní prostředí. Pro začátečníky je doporučeno použít nejprve jednoho z

jednodušších agentů a poté případně zkusit složitější, pokud nebude dosaženo žádaných výsledků. Nejjednodušší agenti vhodní pro diskrétní prostředí, které Matlab nabízí, jsou právě SARSA a Q-learning. Jejich nevýhodou je však absence výchozí hluboké neuronové sítě. Lehce složitější, avšak s možností využití výchozí neuronové sítě je agent DQN (Deep Q-Network). Jedná se vlastně o variantu Q-learningu. DQN agent se nedrží striktně své strategie a k učení využívá dva kritiky. První kritik $Q(S, A; \Phi)$ s parametry Φ přijímá stav S a akci A jako vstupy a vrací kumulativní odměnu. Pro zlepšení stability obsahuje ještě druhého cílového kritika $Q_t(S, A; \Phi_t)$, který je pravidelně aktualizován podle posledních parametrů prvního kritika. Parametry Φ se upravují v průběhu učení a po jeho dokončení jsou uloženy.

Učení DQN agenta probíhá následovně. Nejprve se inicializují parametry kritiků $\Phi = \Phi_t$. Poté se pro současný stav zvolí náhodná akce s pravděpodobností ϵ nebo se zvolí akce, pro kterou je funkce kritika nejvyšší

$$A = arg\max_{A} \left(Q(S, A; \Phi) \right). \tag{4.1}$$

Agent provede zvolenou akci a obdrží odměnu R a další stav S'. Zkušenost S, A, R, S' se uloží do paměti. Pokud je S' finální stav, nastaví se cíl funkce y_i na R_i. Jinak bude stanoven podle vztahu

$$y_i = R_i + \gamma \max_{A'} \left(Q_t(S_i', A'; \Phi_t) \right), \tag{4.2}$$

a kritikovy parametry se aktualizují o ztrátu

$$L = \frac{1}{2M} + \left(\sum_{i=1}^{M} (y_i - Q_t(S_i', A'; \Phi_t) \right)^2. \tag{4.3}$$

Nakonec se aktualizují parametry cílového kritika a pravděpodobnost ϵ. Jak bylo již zmíněno, zpětnovazební učení je velmi výpočetně a taktéž časově náročné. V případě matlabu může učení probíhat od řádově minut až po několik dní v závislosti na komplexnosti problému.[35]

4.3 Návrh modelu pro využití přebytků

Návrh modelového systému pro využití přebytků probíhal podle standardního popsaného postupu. Nejprve je potřeba zvolit prostředí, ve kterém budeme agenta provozovat. Pro lepší názornost jsem se rozhodl prostředí vytvořit v Simulinku, kde budou jasně vidi-

telná jednotlivá propojení oproti čistě textovému prostředí ve formě kódu. Vzhledem k uvažovanému fotovoltaickému systému se bude jednat o využití přebytků ohřevem vody. Základem prostředí je tedy boiler. Bojler je namodelován jako objekt pomocí simscape bloků, které řeší toky tepla. Topné těleso bojleru je napájeno dvěma zdroji. Jedním jsou přebytky fotovoltaického systému, druhým je zdroj o výkonu 2 kW. Oba zdroje tedy dodávají určitý výkon, který ohřívá vodu v bojleru. Ztráty energie jsou způsobeny přenosem tepla z vody na stěnu bojleru a ze stěny do okolí. Parametry pro výpočty jsou v inicializační funkci. Skutečný systém sice využívá dva bojlery, ale simulovat tuto situaci je komplexní záležitost. Museli bychom namodelovat tepelné toky obou bojlerů včetně propojovacího potrubí a tok vody mezi bojlery. To podstatně zvyšuje výpočetní náročnost modelu, a hlavně prodlužuje dobu učení agenta. Ze stejného důvodu je reprezentována spotřeba vody pouze úbytkem energie z bojleru a nikoli fyzickým odtokem vody. Celkové množství energie v bojleru je reprezentováno teplotou vody, kterou měří tepelný senzor.

Hlavním požadavkem vlastníka bojleru, je mít vždy k dispozici teplou vodu. Základním cílem modelu tedy bude udržovat teplotu vody v určitém komfortním rozmezí. Sekundárním cílem je maximalizovat využití přebytků, a tedy minimalizovat spouštění záložního zdroje. To požaduje uživatel kvůli ceně elektřiny ze sítě, která se na ohřev spotřebuje. Toto jsou nejdůležitější požadavky, které byly použity v základním modelu. Model jsem následně zkoušel rozšiřovat o další požadavky, při kterých ale již nebylo dosaženo uspokojivých výsledků. Přesto si myslím, že je důležité i tyto výstupy zmínit, proto budou podrobněji rozebrány v sekci výsledků modelu.

Po zvolení požadavků pro agenta je třeba nadefinovat výpočet odměn. K tomu nejprve potřebujeme stanovit, jaké informace o prostředí bude agent dostávat. V původním modelu jsem jich zvolil 6. Aby mohl udržovat teplotu vody T_W v zadaném rozmezí potřebuje znát tuto teplotu a hranice intervalu T_{min} a T_{max}. Předáme-li agentovi cenu spotřebované elektřiny C, získá tím agent i informaci o množství energie. Kromě energie záložního zdroje samozřejmě musíme agentovi předat informaci o přebytcích fotovoltaického systému. Při rozšiřování modelu jsem přidal další 3 signály. Nejprve byl přidán průběh spotřeby vody. Vzhledem k ztrátám energie v bojleru je vhodné přivést i informaci o okolní teplotě, kterou jsem pro zjednodušení zvolil konstantní. Poslední vstupem je, v jaké hodině dne se právě prostředí nachází.

Výpočet odměn tedy bude probíhat následovně. Agent dostane kladnou odměnu, bude-li udržovat teplotu vody T_W v zadaném rozmezí od T_{min} do T_{max} a zápornou odměnu neboli penalizaci, bude-li teplota mimo rozmezí. Na poloze velikosti teploty v stanoveném intervalu. Budeme uvažovat, že v takovém případě jsou splněny požadavky stejně. Odměna

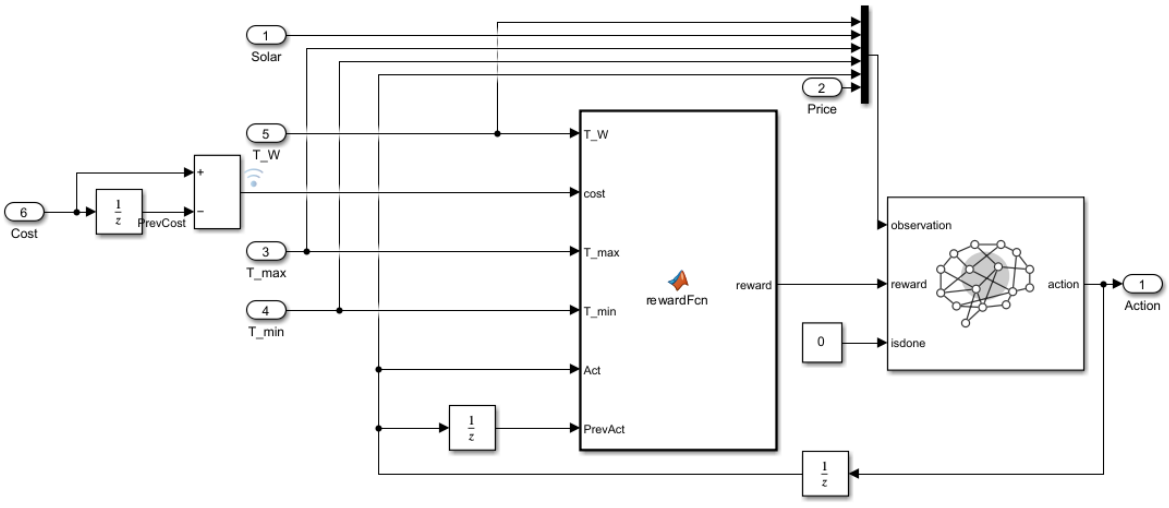
tedy bude pro všechny hodnoty stejná. Čím více se ale odlišuje teplota od požadovaného intervalu, tím je spokojenost uživatele nižší. Velikost penalizace tedy bude záležet na vzdálenosti od intervalu. Ve výsledku bude odměna R_T vypočítána takto:

$$R_T = \begin{cases} 0,5 & \text{pro} \quad T_{min} \leq T_W \leq T_{max} \\ -0,5|T_W - T_{min}| & \text{pro} \quad T_W \leq T_{min} \\ -0,5|T_{max} - T_W| & \text{pro} \quad T_W \geq T_{min}. \end{cases} \tag{4.4}$$

Stanovení výše odměny není zcela jednoznačný proces, spíše se jedná o odhad, který se následně ladí při učení agenta. Zbylé odměny jsou definovány obdobně. Penalizace za provoz záložního zdroje R_C je vypočtena přenásobením ceny spotřebované energie C konstantou.

$$R_C = -0,01 * C \tag{4.5}$$

Aby se záložní zdroj spouštěl co nejméně, je agent navíc penalizován i za každé sepnutí. To je realizováno srovnáním současné a předcházející akce. V případě, že jsou odlišné, dostane agent odměnu -0,01. Pokud jsou obě akce stejné, je odměna nulová. Výsledný subsystém s funkcí odměn a blokem agenta je na obr. 4.4.



Obrázek 4.4: Subsystém zpětnovazebního učení v prostředí simulink

V prostředí simulinku je agent reprezentován blokem RL agent. Samotný blok ale ve výchozím nastavení agenta neobsahuje a je nutné ho vytvořit pomocí skriptu. Nejprve se specifikují základní možnosti agenta.

```
%specifikace stavu a akci
obsInfo = rlNumericSpec([6,1]);
actInfo = rlFiniteSetSpec([0,1]);

%parametry uceni kritika
criticOpts = rlOptimizerOptions( ...
    LearnRate=0.001, ...
    GradientThreshold=1);

%parametry agenta
agentOpts = rlDQNAgentOptions(...
    UseDoubleDQN = false, ...
    TargetSmoothFactor = 1, ...
    TargetUpdateFrequency = 4, ...
    ExperienceBufferLength = 1e6, ...
    CriticOptimizerOptions = criticOpts, ...
    MiniBatchSize = 64);
```

Příkaz rlNumericSpec se využívá pro kontinuální prostředí a specifikuje dimenzi vektoru stavů. Druhý řádek určuje přímo diskrétní hodnoty akce agenta. Akce 1 znamená sepnutí záložního zdroje, akce 0 vypnutí. V druhém odstavci se nastavují vlastnosti kritika. Specificky se jedná o rychlost učení a limit gradientu. V případě vyššího gradientu je současný gradient snížen na hranici limitu, aby nedocházelo k příliš velkým změnám v daném kroku. Z dalších parametrů je zajímavá velikost bufferu zkušeností, kam se ukládá průběh učení agenta. Pokud chceme pokračovat v učení již trénovaného agenta, musíme tento buffer uložit. Podstatnou nevýhodou je však vysoká paměťová náročnost tohoto bufferu.

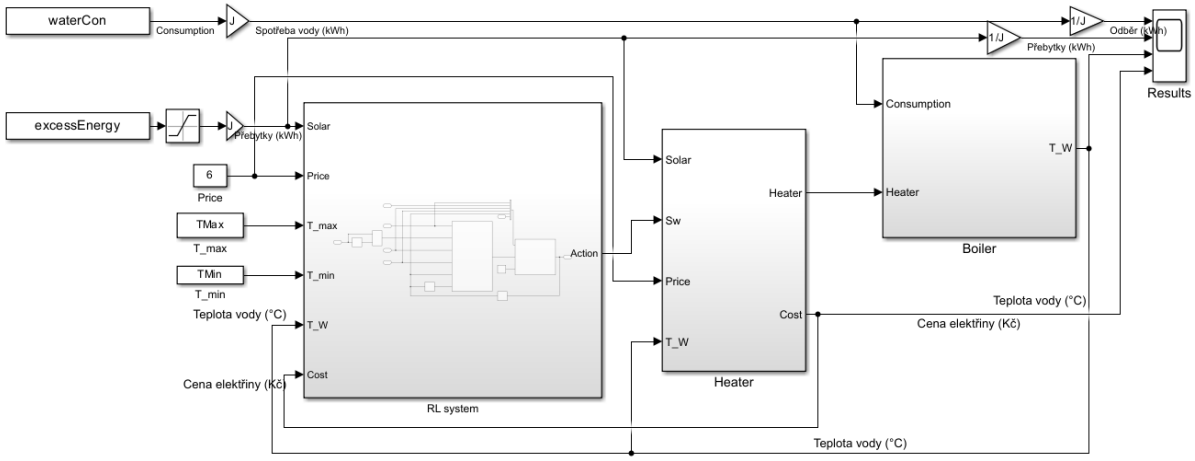
Pro realizaci strategie agenta je použita rekurentní LSTM (Long short-term memory) neuronová síť s 64 neurony v každé skryté vrstvě. Název "dlouhodobá krátkodobá paměť"se může zdát zavádějící, ale opravdu popisuje princip sítě. Síť totiž ukládá informace o předcházejícím stavu (krátkodobá paměť), které ale následně musí mít uložené po velké množství následujících kroků (dlouhodobá pamět). Výsledkem je neuronová síť, která se dokáže naučit dlouhodobé závislosti v časové posloupnosti dat. Tato síť je posledním parametrem potřebným k vytvoření samotného agenta. Po vytvoření agenta můžeme nadefinovat a ověřit prostředí. Vše je realizováno následujícím kódem:

```
%volba neuronove site
useRNN = true;
initOpts = rlAgentInitializationOptions( ...
    UseRNN=useRNN, ...
    NumHiddenUnit=64);

$vytvoreni objektu agenta
agent = rlDQNAgent(obsInfo, actInfo, initOpts, agentOpts);

$tvorba a overeni prostredi
agentBlk = mdl + "/RL system/RL Agent";
env = rlSimulinkEnv(mdl,agentBlk,obsInfo,actInfo);
env.ResetFcn = @(in) BResetFcn(in);
validateEnvironment(env)
```

Jakmile je definován agent i jeho prostředí, přichází na řadu již samotné učení. Při učení se simuluje celé prostředí, ve kterém agent provádí akce a dostává odměny. Jeden cyklus učení se nazývá epizodou. Nejdůležitější parametry, které se specifikují před začátkem učení je maximální počet epizod a kroků za epizodu a ukončovací kritérium. Učení agenta skončí, jestliže je dosaženo ukončovacího kritéria nebo dosaženo maximálního počtu epizod. V tomto případě bylo jako ukončovací kritérium použita hodnota odměny 50, jak je vidět ve výstřižku z kódu.

```
trainOpts = rlTrainingOptions(...
    MaxEpisodes = 150, ...
    MaxStepsPerEpisode = 1000, ...
    ScoreAveragingWindowLength = 5,...
    Verbose = false, ...
    Plots = "training-progress",...
    StopTrainingCriteria = "AverageReward",...
    StopTrainingValue = 50);

trainingStats = train(agent,env,trainOpts);
```

Celkové schéma modelu v prostředí simulink je na obr. 4.5. Jednotlivé substémy naleznete v příloze.



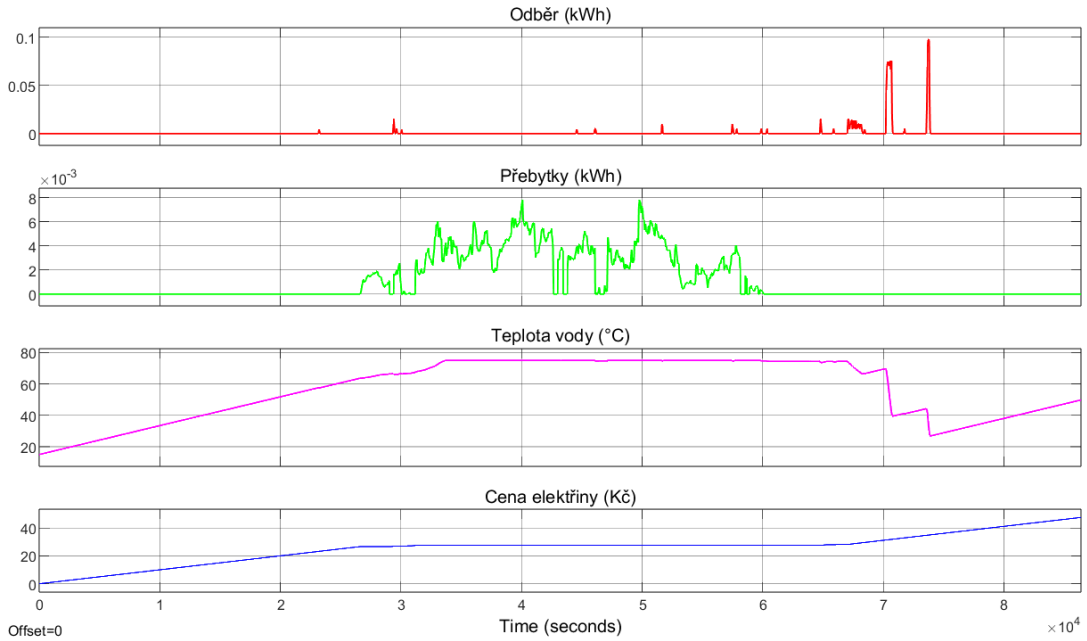
Obrázek 4.5: Celkový model pro řízení ohřevu vody

4.4 Výsledky modelu a diskuze

Základní model byl trénován na simulovaných i naměřených datech. Výsledky trénování byly pro obě sady daty prakticky totožné. V obou případech se agent naučil provozovat systém podle základních požadavků. Doba trénování se pohybovala kolem 2 hodin. Průběh jednoho z úspěšných trénování naleznete v příloze.

Po naučení agenta byla funkce agenta ověřena na skutečných naměřených datech zvoleného domácího fotovoltaického systému v Řevnici z dubna roku 2023. Na obr. 4.6 jsou průběhy sledovaných veličin v období celého dne 1. dubna 2023. Uvažujeme, že přesně o půlnoci byl systém uveden do provozu s počáteční teplotou 15 ° C. Zpočátku dne neprodukuje fotovoltaika žádnou energii, proto je neustále sepnut záložní zdroj, dokud není dosaženo požadované teploty. Ráno začne fotovoltaika produkovat a ohřívat kotel, což způsobí vypnutí záložního zdroje. V průběhu dne, kdy není velká spotřeba vody a je velké množství přebytků, je teplota v kotli udržována téměř pouze přebytky. Agenta tedy nechává po většinu doby záložní zdroj vypnutý a cena spotřebované elektřiny tedy víceméně nestoupá. Změna nastává večer, kdy už nejsou přebytky z fotovoltaiky a navíc dojde ke dvou velkým odběrům teplé vody. To způsobí velký pokles energie v bojleru, který agent kompenzuje sepnutím záložního zdroje. Agent tedy plní svou funkci udržování teplotního rozmezí v kotli.

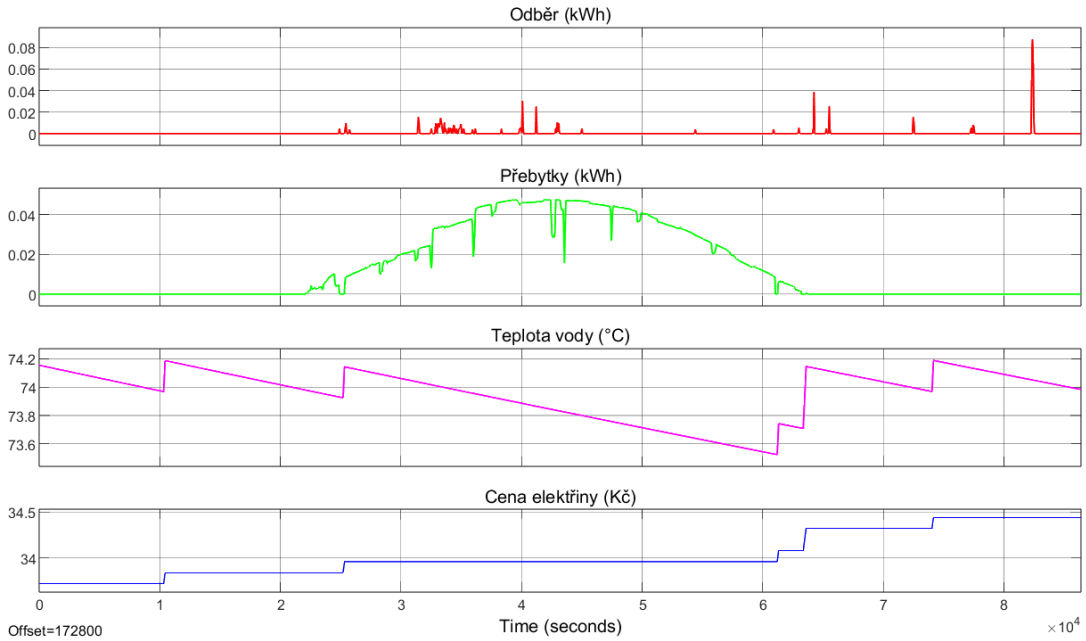
Pro lepší viditelnost funkce agenta jsem provedl jednu simulaci, kdy jsem odpojil přebytky a odběr vody od bojleru a nechal vodu ohřívat pouze záložním zdrojem. Zachoval jsem ale připojení informace o přebytcích agentovi. Tím jsem získal průběh na



Obrázek 4.6: Simulace agenta na datech fotovoltaického systému 1. dubna 2023

obr. 4.7. Na první pohled je vidět, že spínání agenta není optimální. na druhou stranu je vidět, že v období přebytků záložní zdroj nespíná, sepne ho pouze ve dvou bodech, kdy přebytky klesnou k nule. Agent tedy stále využívá informaci o přebytcích, přestože nejsou využívány k ohřevu.

Tento základní model jsem zkoušel vylepšit mnoha způsoby. Původní myšlenka byla, aby byl agent schopen přepínat nezávisle mezi přebytky a záložním zdrojem. Ale již tato zdánlivě jednoduchá úprava vedla k několikanásobnému prodloužení doby učení (v průměru kolem 9 hodin), které nikdy nevedlo ke kladné celkové odměně. Proto jsem se rozhodl zachovat akci agenta na přepínání záložního zdroje a pokusil se upravit parametry prostředí a stavy, které agent dostává. Záměrem bylo, aby se agent naučil vzor průběhů produkce a spotřeby a předvídal v jakém denním období tedy k přebytkům a odběrům dochází, a přizpůsobil tak předehřev vody. Bohužel i toto vedlo k neschopnosti agenta dosáhnout kladné odměny v průběhu několikahodinového učení. Určení problému není zcela přímočaré. Samozřejmě se nabízí možnost nedostatečného výpočetního výkonu a nedostatečné doby učení. Toto mi nejprve nedávalo smysl, jelikož jsem předpokládal, že v takovém případě by musela být v průběhu učení patrná konvergence odměny. Jak si můžete všimnout na obr. 4.8, odměna se sice během prvních pár epizod zvyšovala, ale následně stagnovala a agent nebyl schopen nalézt lepší řešení. V dalších epizodách docházelo k extrémním fluktuacím výše odměny, kdy se agent pravděpodobně snažil silně odchýlit od svého předcházejícího postupu ve snaze nalézt alternativní řešení problému. Hodnota celkové odměny tedy s výjimkou prvních pár epizod patrně nekonvergovala. Kon-

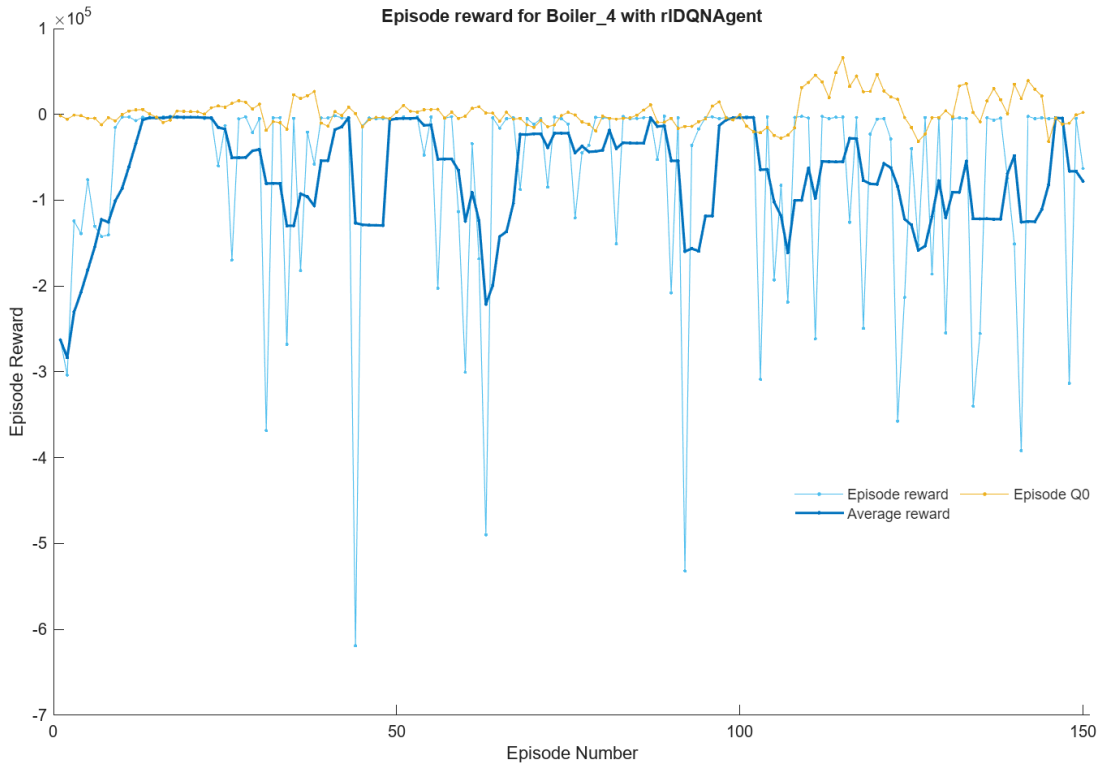


Obrázek 4.7: Ukázka udržování teploty agentem

vergence nenastávala ani mezi velikosti odměny a velikostí Q0, což je maximální kritikem předpovězená diskontovaná odměna na začátku každé epizody. V případě agentů, které mají pouze kritika je shoda Q0 je konvergence těchto dvou hodnot znamením úspěšného učení. Později jsem ale nalezl například model [36], kde taktéž docházelo k extrémním fluktuacím odměny a nejprve dokonce odměna klesala a vzdalovala se od hodnoty Q0. V určitém bodě ale nastal zlom a obě hodnoty skonvergovaly. Komplexnější návrhy řízení ohřevu vody by tedy mohly eventuálně taktéž konvergovat, ale agent určený k použití v domácím fotovoltaickém systému, který vyžaduje několikadenní učení určitě není vhodnou alternativou k wattrouteru či jiným regulačním zařízením.

Samozřejmě průběh učení závisí na mnoha faktorech a určitě by bylo možné model lépe optimalizovat. Například volbou jiné strategie agenta. Rekurentní neuronové sítě s dlouhodobou pamětí jsou sice velmi vhodné pro tento záměr, jelikož umožňují nalezení dlouhodobých vztahů v časových posloupnostech, ale na druhou stranu jsou výpočetně náročnější než klasické dopředné sítě. Nabízí se i možnost vytvoření vlastní neuronové sítě, která bude navržena přímo pro daný problém. To ale limituje univerzálnost agenta a případná rozšíření řídícího algoritmu. Optimalizace zpětnovazebního učení je komplexní záležitostí, která vyžaduje spoustu zkušeností, což je další nevýhodou nástrojů umělé inteligence. Ve většině případů je vytvoření funkčního systému na bázi strojového učení složitější než využití standardně používaných nástrojů, které jsou uživatelsky přívětivější. Software Matlab sice nabízí obrovské množství možností, což je ale pro začátečníka spíše na škodu. Z hlediska univerzálnosti nemají umělé neuronové sítě konkurenci, ale z hlediska

praktičnosti je ještě čeká dlouhá cesta.



Obrázek 4.8: Průběh neúspěšného učení agenta

Praktická implementace řízení pomocí zpětnovazebního učení v této práci sice neměla výrazný úspěch, ale naštěstí lze nalézt mnoho studií, které potenciál této technologie ukazují. V současnosti se hodně zaměřují na využití cloudů neboli učení umělých neuronových sítí či agentů dálkově pomocí serverových počítačů. Existence většího množství prvků například v komunitě vlastníků fotovoltaických systémů umožňuje podstatně efektivnější řízení pomocí specializovaných výpočetních jednotek. Zajímavých výsledků bylo dosaženo například ve studii [37], kde vytvořili řídící síť ohřevu vody v 50 domácnostech pomocí kolaborace většího množství agentů. V takové sítí může každý agent řídit svou domácnost podle individuálních preferencí a zároveň se přizpůsobovat komunitním požadavkům díky komunikaci s ostatními agenty. Podle studie je tento přístup několikanásobně efektivnější než systém s jedním agentem. To dokazují ušetřením téměř 200 kWh energie po dobu jednoho roku pro každou domácnost. Z výsledků studie tedy vyplývá, že ve velkém měřítku určitě tato technologie potenciál má.

Závěr

Cílem práce bylo ukázat potenciál umělé inteligence a případné nedostatky využití umělé inteligence pro řízení energetických systémů. V rešeršní části bylo předvedeno, že nástroje strojového učení a zejména umělé neuronové sítě naleznou v oblasti mikrosítí a virtuálních elektráren rozsáhlé využití a jsou důležitým prvkem přechodu k chytrým sítím. Jejich rozšíření však zatím limituje složitost realizace.

Z hlediska praktické implementace byl ukázán návrh nástrojů umělé inteligence v statickém a dynamickém prostředí. Pro statické prostředí byla použitá mělká neuronová síť, jejímž cílem bylo nalézt vztah mezi dvěma soubory dat pro optimalizaci fotovoltaické elektrárny. V tomto ohledu se neuronová síť ukázala jako velmi účinný a rychlý nástroj. Horších výsledků bylo dosaženo u dynamického prostředí, pro které byl zvolen fotovoltaický systém s ukládáním přebytků do vody a záložním kotlem. Řízení bylo provedeno pomocí zpětnovazebního učení, kdy agent vykonává akce, za které je odměňován. Odměny jsou nastaveny tak, aby maximální výše odměny odpovídala optimálnímu provozu systému. Agent byl schopen naučit se základní požadavky řízení, ale nedokázal zpracovat žádnou komplexnější úlohu.

Zpětnovazební učení se na první pohled zdá jako perfektní možnost pro řízení fotovoltaických systémů díky své univerzálnosti. Uživateli umožňuje stanovení prakticky libovolných požadavků, podle vlastních specifických potřeb. Jak se ale potvrdilo, podstatným nedostatkem je výpočetní náročnost této metody a poměrně složitá implementace. Pro soukromé využití v domácnosti tedy tento způsob řízení fotovoltaických systémů s akumulací doporučit nemohu. Na druhou stranu se tyto technologie v současnosti díky své popularitě vyvíjejí extrémní rychlostí. Lze tedy očekávat, že postupně dojde k lepší optimalizaci a možnosti provozovat komplexnější algoritmy i na méně výkonných jednotkách. Společně s neustálým vývojem výpočetních technologií se tato metoda jeví jako velmi perspektivní.

Na práci lze navázat v mnoha ohledech. V práci je popsán postup vytvoření umělé neuronové sítě a realizaci zpětnovazebního učení v softwaru Matlab. Přestože výsledky řízení přebytků nebyly uspokojivé, určitě je možné dosáhnout lepší optimalizace a implementovat komplexnější způsoby. V rámci práce je zmíněno několik návrhů na rozšíření zaměřených zejména na individuální potřeby uživatele a využití prediktivních vlastností umělé neuronové sítě.

Přílohy

Příloha A

Kód pro vytvoření mělké neuronové sítě

```
%import dat
clear;
data=readmatrix('summary_half.xlsx');
xr=data(:,1);
yr=data(:,2:13);

%Trenovani siti s ruznymi pocty neuronu skryte vrstvy
xt=xr';
yt=yr';
for i = 1:50
    hlayersize = i;
    net = fitnet(hlayersize);
    net.divideParam.trainRatio=60/100;
    net.divideParam.valRatio=30/100;
    net.divideParam.testRatio=10/100;
    [net,tr]= train(net,xt,yt);

    yTrain = net(xt(:,tr.trainInd));
    yTrainTrue = yt(:,tr.trainInd);
    yVal = net(xt(:,tr.valInd));
    yValTrue = yt(:,tr.valInd);
    yTest = net(xt(:,tr.testInd));
    yTestTrue = yt(:,tr.testInd);
    rmse_train(i) = sqrt(mean((yTrain-yTrainTrue).^2,"all"));
```

```
    rmse_val(i) = sqrt(mean((yVal-yValTrue).^2,"all"));
    rmse_test(i) = sqrt(mean((yTest-yTestTrue).^2,"all"));
end

%Nalezeni optimalniho poctu neuronu skryte site
plot(1:50,rmse_train);
plot(1:50,rmse_val);
plot(1:50,rmse_test);
[M1,I1] = min(rmse_train)
[M2,I2] = min(rmse_val)
[M3,I3] = min(rmse_test)
[M,I] = min(rmse_train+rmse_val+rmse_test)

%Trenovani optimalni neuronove site
hlayersize = 3;
net = fitnet(hlayersize);
net.divideParam.trainRatio=60/100;
net.divideParam.valRatio=30/100;
net.divideParam.testRatio=10/100;
[net,tr]= train(net,xt,yt);

%Kontrola chyby optimalni neuronove site
yTrain = net(xt(:,tr.trainInd));
yTrainTrue = yt(:,tr.trainInd);
yVal = net(xt(:,tr.valInd));
yValTrue = yt(:,tr.valInd);
yTest = net(xt(:,tr.testInd));
yTestTrue = yt(:,tr.testInd);

rrmse_train = sqrt(mean((yTrain-yTrainTrue).^2,"all"))
rrmse_val = sqrt(mean((yVal-yValTrue).^2,"all"))
rrmse_test = sqrt(mean((yTest-yTestTrue).^2,"all"))

%Graficke srovnani vystupu site s realnymi daty
plot(yTrainTrue,yTrain,'.')
plot(yValTrue,yVal,'.')
plot(yTestTrue,yTest,'.')
```

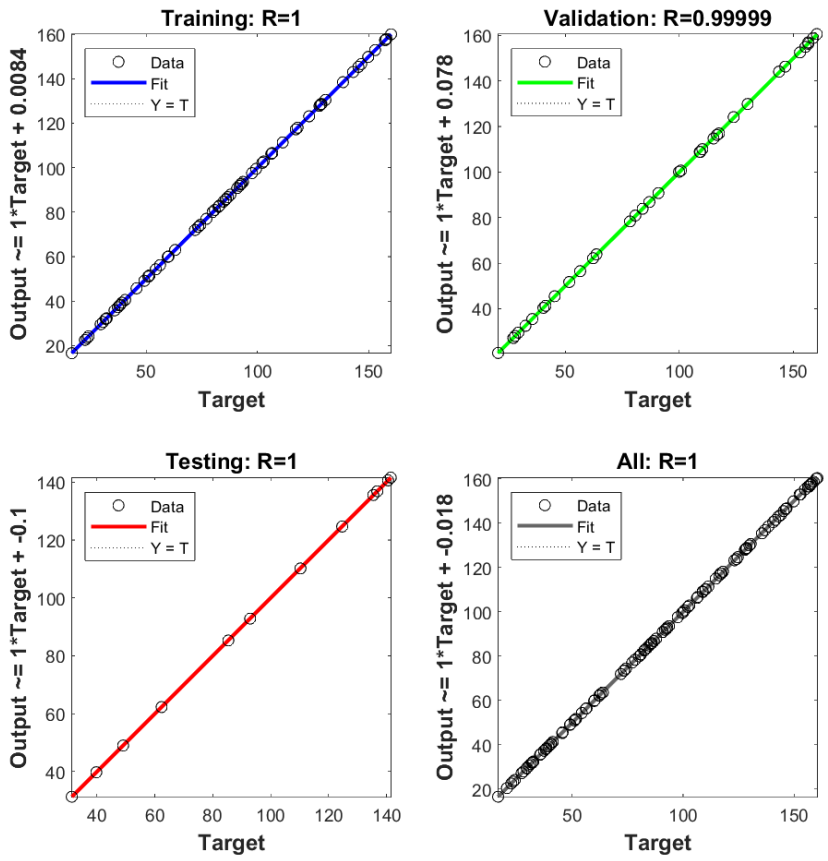
```
%%
plot((1:12),yTrainTrue)
axis([0 20 0 200])
legend(string(xt(1,tr.trainInd)))

plot((1:12),yTrain)
axis([0 20 0 200])
legend(string(xt(1,tr.trainInd)))

plot((1:12),yValTrue)
axis([0 20 0 200])
legend(string(xt(1,tr.valInd)))

plot((1:12),yVal)
axis([0 20 0 200])
legend(string(xt(1,tr.valInd)))

plot((1:12),yTestTrue)
axis([0 20 0 200])
legend(string(xt(1,tr.testInd)))

plot((1:12),yTest)
axis([0 20 0 200])
legend(string(xt(1,tr.testInd)))
%%
for i = 1:length(tr.trainInd)
plot((1:12),[yTrain(:,i),yt(:,tr.trainInd(i))],'.')
axis([0 15 0 200])
legend(string(xt(1,tr.trainInd(i))) + ' PVsol',string(xt(1,tr
   .trainInd(i))) + ' ANN')
figure()
end
%%
for i = 1:length(tr.valInd)
plot((1:12),[yVal(:,i),yt(:,tr.valInd(i))],'.')
axis([0 15 0 200])
legend(string(xt(1,tr.valInd(i))) + ' PVsol',string(xt(1,tr.
```

```
   valInd(i))) + ' ANN')
figure()
end
%%
for i = 1:length(tr.testInd)
plot((1:12),[yTest(:,i),yt(:,tr.testInd(i))],'.')
axis([0 15 0 200])
legend(string(xt(1,tr.testInd(i))) + ' PVsol',string(xt(1,tr.
   testInd(i))) + ' ANN')
figure()
end
%%
%Parametry neuronove site
y = net(xt);
e = gsubtract(yt,y);
performance = perform(net,yt,y)
%%
view(net)
%%
%Parametry presnosti a kvality neuronove site
figure, plotperform(tr)
figure, plottrainstate(tr)
figure, ploterrhist(e)
figure, plotregression(yTrain,yTrainTrue,'Training',yVal,
   yValTrue,'Validation',yTest,yTestTrue,'Testing',yt,y,'All'
   )
figure, plotfit(net,xt,yt)
%%
%Kontrola neuronove site na externich datech
extdata=readmatrix('summary_half_2.xlsx');
extx=extdata(:,1)';
exty=extdata(:,2:13)';
%%
extnety=net(testx)
rmse_ext = sqrt(mean((extnety-exty).^2,"all"))
%%
for i = 1:length(extx)
```

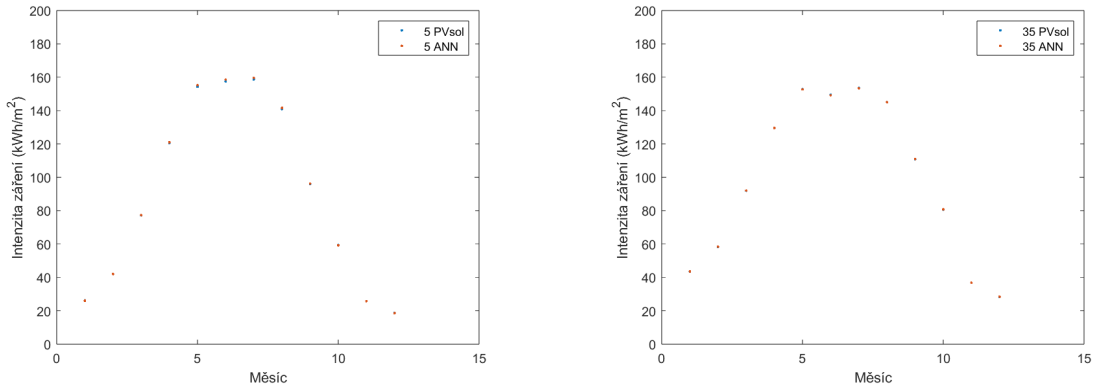
```
plot((1:12),[extnety(:,i),exty(:,i)],'.')
axis([0 15 0 200])
legend(string(extx(i)) + ' PVsol',string(extx(i)) + ' ANN')
figure()
end
%%
%Sklon pro maximalni rocni produkci
[prod,index]=max(sum(net(1:90)))
```

Příloha B

Výstupy neuronové sítě



Obrázek B.1: Srovnání výstupů sítě se skutečnými výstupy



(a) Sklon 5 ° (b) Sklon 35 °

Obrázek B.2: Srovnání skutečných hodnot s výstupem neuronové sítě

Příloha C

Kód pro vytvoření agenta zpětnovazebního učení

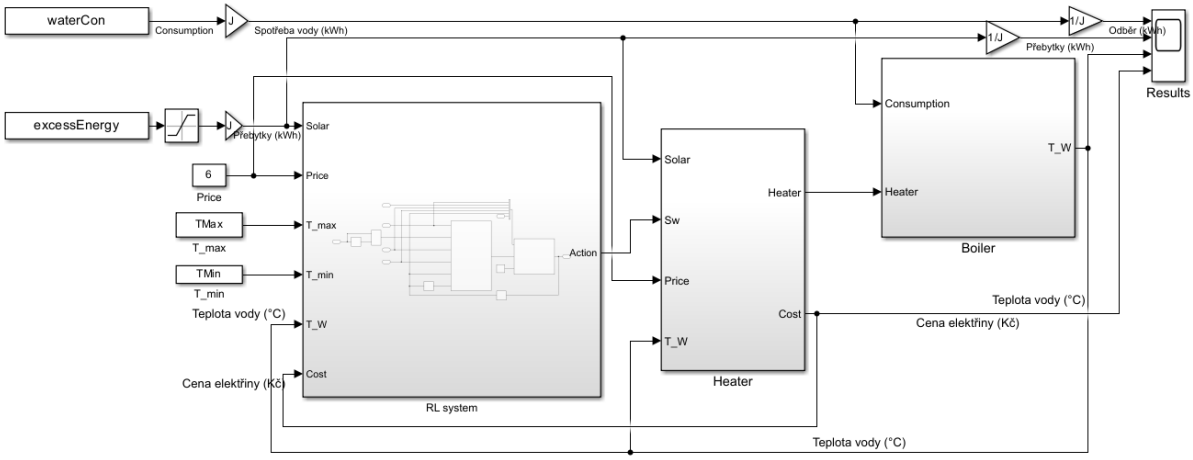
```
% import dat
clear;
data=readmatrix('Revnice_data.xlsx');
hr=data(1:60*24*30,[1,2]);
excessEnergy=data(1:60*24*30,[1,3]);
waterCon=data(1:60*24*30,[1,4]);

% Sumulink model
mdl = "excessEnControl_v1b";
open_system(mdl)

% specifikace zakladnich parametru
sampleTime = 120;
maxStepsPerEpisode = 1000;
agentBlk = mdl + "/RL system/RL Agent";
comfortMax = 75;
comfortMin = 68;
obsInfo = rlNumericSpec([6,1]);
actInfo = rlFiniteSetSpec([0,1]);
rng(0)

% Vytvoreni DQN agenta
criticOpts = rlOptimizerOptions( ...
    LearnRate=1, ...
```

```
    GradientThreshold=1);
agentOpts = rlDQNAgentOptions(...
    UseDoubleDQN = false, ...
    TargetSmoothFactor = 0.1, ...
    TargetUpdateFrequency = 4, ...
    ExperienceBufferLength = 1e6, ...
    CriticOptimizerOptions = criticOpts, ...
    MiniBatchSize = 64);
agentOpts.EpsilonGreedyExploration.EpsilonDecay = 0.0001;
agentOpts.SequenceLength = 20;

%volba neuronove site
useRNN = true;
initOpts = rlAgentInitializationOptions( ...
    UseRNN=useRNN, ...
    NumHiddenUnit=64);

%
agent = rlDQNAgent(obsInfo, actInfo, initOpts, agentOpts);
agent.SampleTime = sampleTime;

% Definice prostredi
env = rlSimulinkEnv(mdl,agentBlk,obsInfo,actInfo);
env.ResetFcn = @(in) BResetFcn(in);
validateEnvironment(env)
% Trenovani agenta
trainOpts = rlTrainingOptions(...
    MaxEpisodes = 150, ...
    MaxStepsPerEpisode = 1000, ...
    ScoreAveragingWindowLength = 5,...
    Verbose = false, ...
    Plots = "training-progress",...
    StopTrainingCriteria = "AverageReward",...
    StopTrainingValue = 50);

%vyber trenovani nebo jiz nauceneho agenta
doTraining = false;
```

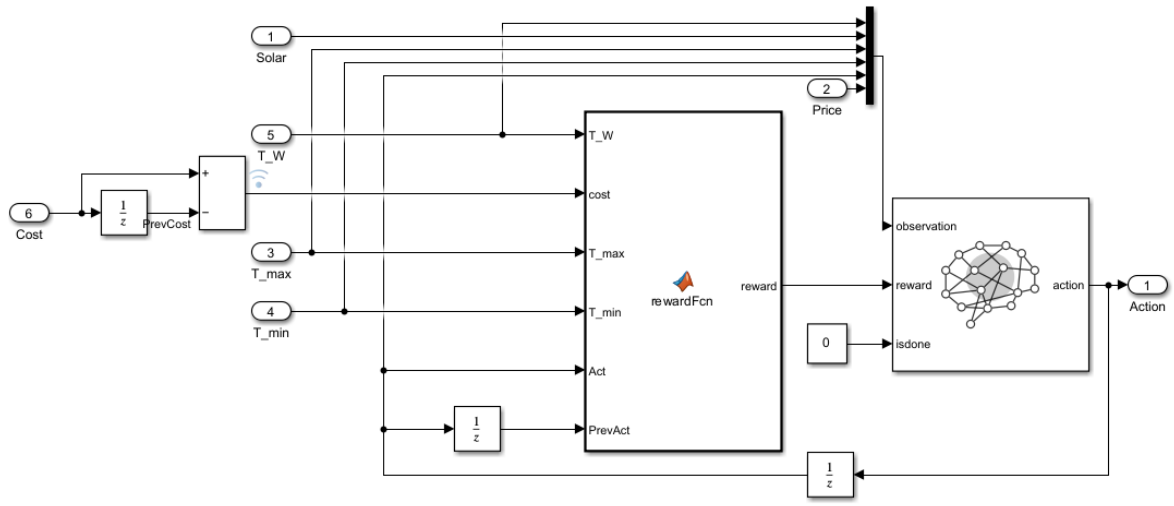
```
if doTraining
    trainingStats = train(agent,env,trainOpts);
else
    load("ExcessEnControlDQNAgent.mat","agent")
end
```

Příloha D

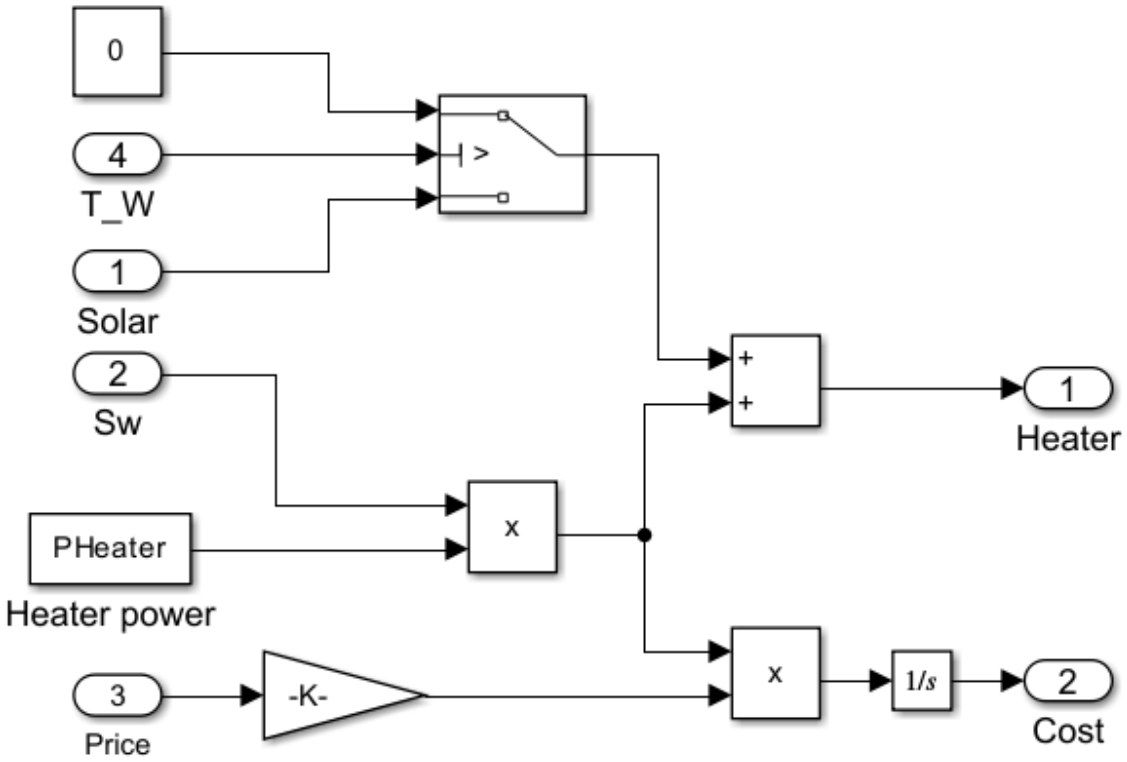
Model využití přebytků v simulinku



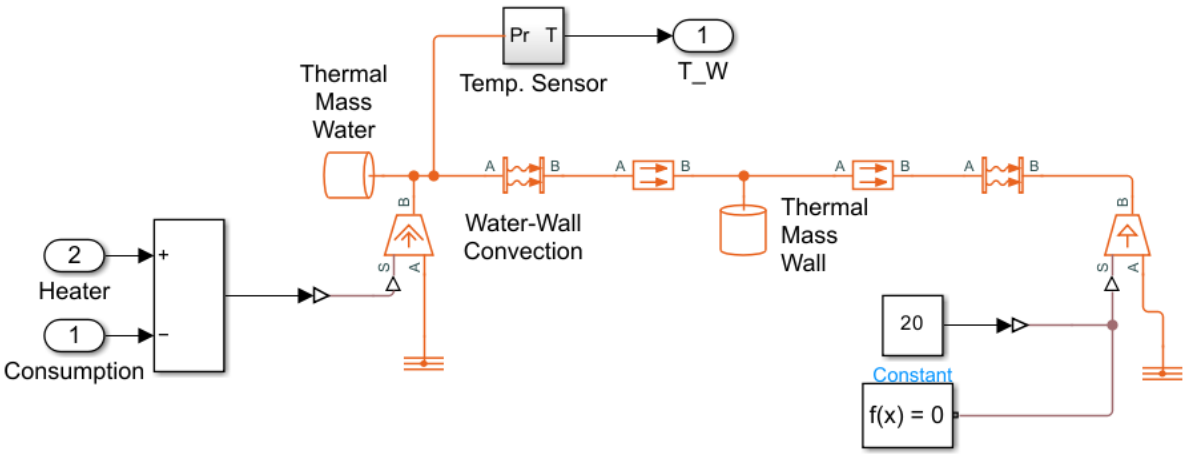
Obrázek D.1: Celkový model pro řízení ohřevu vody



Obrázek D.2: Subsystém zpětnovazebního učení v prostředí simulink



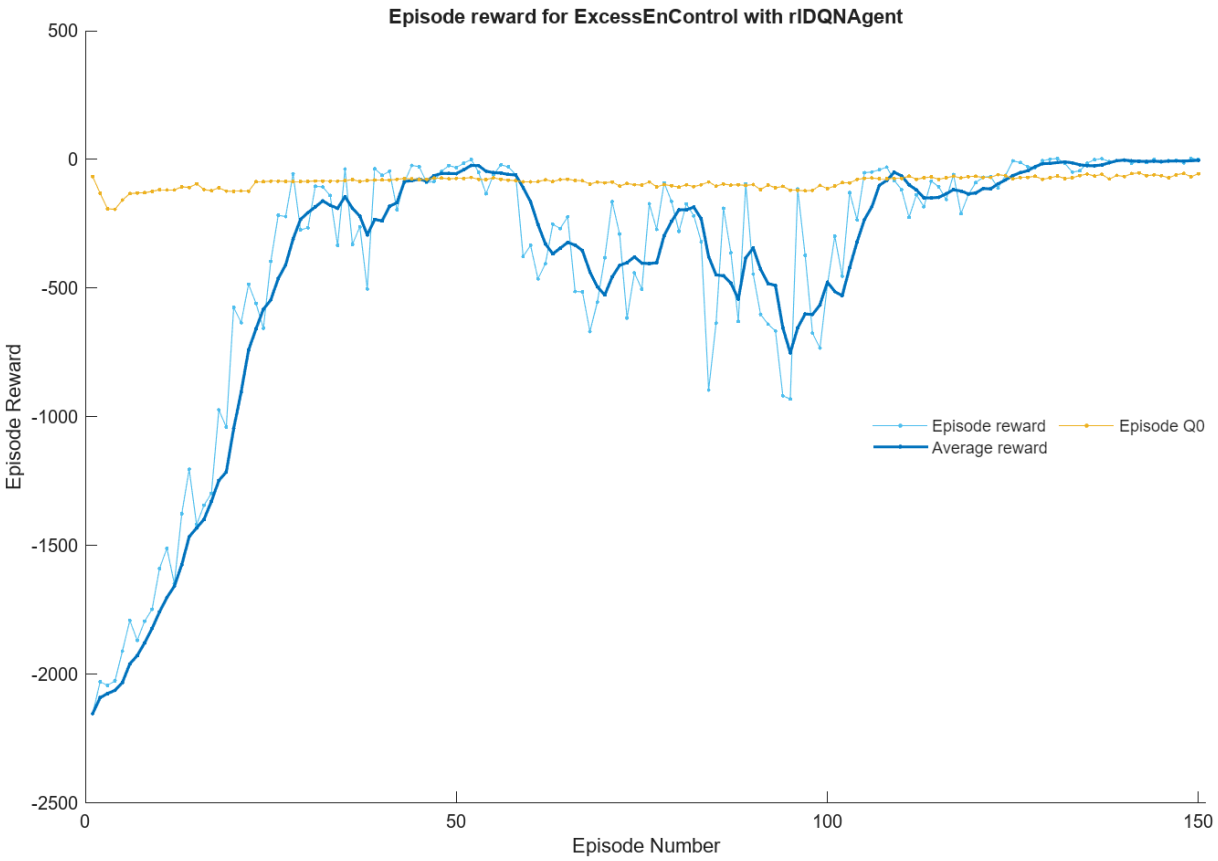
Obrázek D.3: Subsystém ohřevu



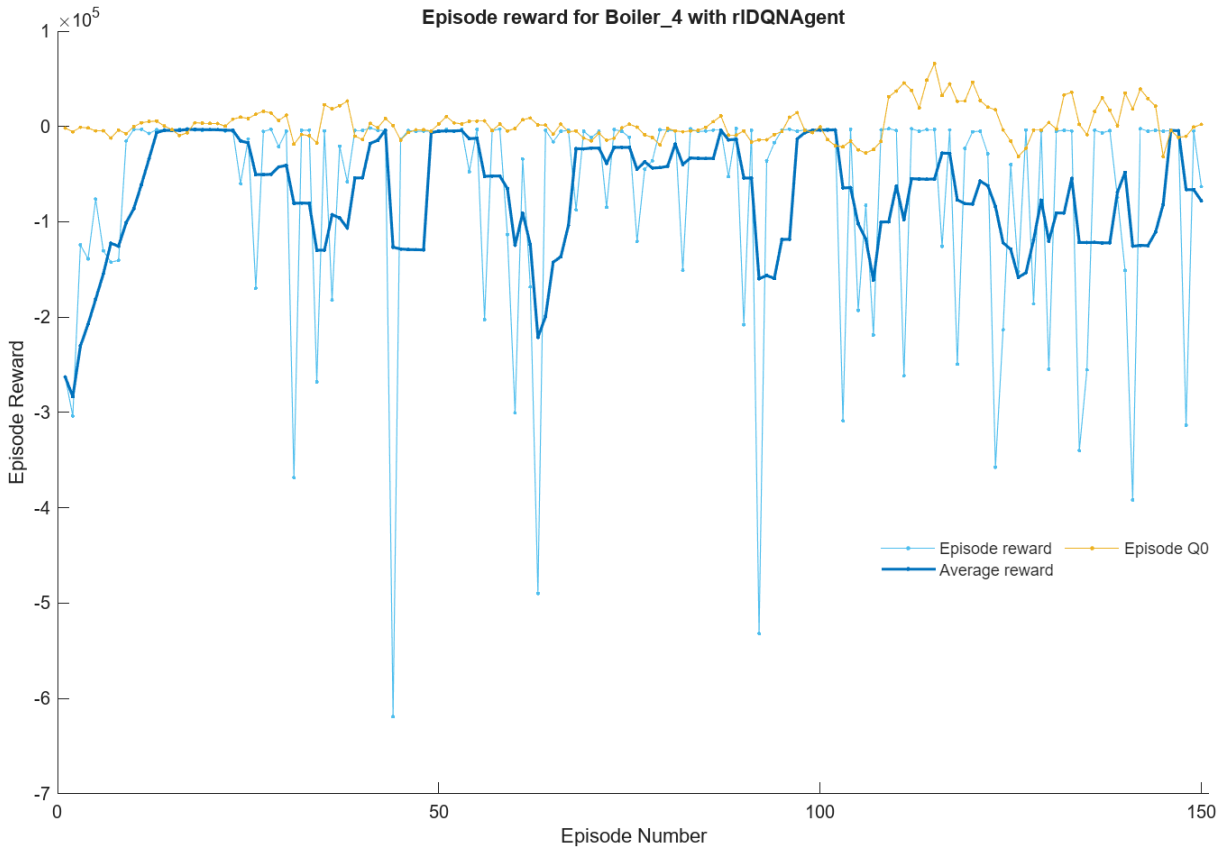
Obrázek D.4: Subsystém bojleru

Příloha E

Průběh učení agenta



Obrázek E.1: Úspěšné učení agenta



Obrázek E.2: Neúpěšné učení agenta

Literatura

[1] L. Mariam, M. Basu a M. F. Conlon, "Microgrid: Architecture, policy and future trends", *Renewable and Sustainable Energy Reviews*, roč. 64, s. 477–489, 2016, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2016.06.037. URL: https://www.sciencedirect.com/science/article/pii/S1364032116302635.

[2] E. Planas, J. Andreu, J. I. Gárate, I. Martínez de Alegría a E. Ibarra, "AC and DC technology in microgrids: A review", *Renewable and Sustainable Energy Reviews*, roč. 43, s. 726–749, 2015, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2014.11.067. URL: https://www.sciencedirect.com/science/article/pii/S1364032114010065.

[3] F. Ferrigolo, D. Ramos, J. Correa, F. Farret a L. Porto, "Advanced High Frequency AC Microgrid with integrated power quality conditioning capability", s. 178 –183, pros. 2009. DOI: 10.1109/IECON.2009.5414767.

[4] S. Sen a V. Kumar, "Microgrid control: A comprehensive survey", *Annual Reviews in Control*, roč. 45, s. 118–151, 2018, ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2018.04.012. URL: https://www.sciencedirect.com/science/article/pii/S1367578818300373.

[5] R. Trivedi a S. Khadem, "Implementation of artificial intelligence techniques in microgrid control environment: Current progress and future scopes", *Energy and AI*, roč. 8, s. 100 147, 2022, ISSN: 2666-5468. DOI: https://doi.org/10.1016/j.egyai.2022.100147. URL: https://www.sciencedirect.com/science/article/pii/S266654682200009X.

[6] M. Hagan, H. Demuth, M. Beale a O. De Jesús, *Neural Network Design*. Martin Hagan, 2014, ISBN: 9780971732117. URL: https://books.google.cz/books?id=4EW9oQEACAAJ.

[7] M. Q. Raza a A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings", *Renewable and Sustainable Energy Reviews*, roč. 50, s. 1352–1372, 2015, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.04.065. URL: https://www.sciencedirect.com/science/article/pii/S1364032115003354.

[8] C. Watkins a P. Dayan, "Technical Note: Q-Learning", *Machine Learning*, roč. 8, s. 279–292, květ. 1992. DOI: 10.1007/BF00992698.

[9] Y. Li, "Deep Reinforcement Learning: An Overview", *CoRR*, roč. abs/1701.07274, 2017. arXiv: 1701.07274. URL: http://arxiv.org/abs/1701.07274.

[10] E. Mocanu, P. H. Nguyen, W. L. Kling a M. Gibescu, "Unsupervised energy prediction in a Smart Grid context using reinforcement cross-building transfer learning", *Energy and Buildings*, roč. 116, s. 646–655, 2016, ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2016.01.030. URL: https://www.sciencedirect.com/science/article/pii/S0378778816300305.

[11] J.-S. Ko, J.-H. Huh a J.-C. Kim, "Overview of Maximum Power Point Tracking Methods for PV System in Micro Grid", *Electronics*, roč. 9, s. 816, květ. 2020. DOI: 10.3390/electronics9050816.

[12] N. Chettibi a A. Mellit, "Intelligent control strategy for a grid connected PV/SOFC/BESS energy generation system", *Energy*, roč. 147, s. 239–262, břez. 2018. DOI: 10.1016/j.energy.2018.01.030.

[13] N. Chettibi, A. Mellit, G. Sulligoi a A. Massi Pavan, "Adaptive Neural Network-Based Control of a Hybrid AC/DC Microgrid", *IEEE Transactions on Smart Grid*, roč. PP, s. 1–1, srp. 2016. DOI: 10.1109/TSG.2016.2597006.

[14] M. Jafari, V. Sarfi, A. Ghasemkhani et al., "Adaptive neural network based intelligent secondary control for microgrids", s. 1–6, ún. 2018. DOI: 10.1109/TPEC.2018.8312064.

[15] M. Bagheri, V. Nurmanova, O. Abedinia a M. Salay Naderi, "Enhancing Power Quality in Microgrids with a New Online Control Strategy for DSTATCOM Using Reinforcement Learning Algorithm", *IEEE Access*, roč. PP, s. 1–1, čvc. 2018. DOI: 10.1109/ACCESS.2018.2852941.

[16] V. Awasth a V. Huchche, "Reactive power compensation using D-STATCOM", s. 583–585, dub. 2016. DOI: 10.1109/ICEETS.2016.7583821.

[17] R. Heydari, Y. Khayat, A. Amiri et al., "Robust High-Rate Secondary Control of Microgrids With Mitigation of Communication Impairments", *IEEE Transactions on Power Electronics*, roč. PP, s. 1–1, dub. 2020. DOI: 10.1109/TPEL.2020.2986368.

[18] H. Lin, K. Sun, Z.-H. Tan, C. Liu, J. Guerrero a J. C. Vasquez, "Adaptive Protection Combined with Machine Learning for Microgrids", *IET Generation, Transmission & Distribution*, roč. 13, břez. 2019. DOI: 10.1049/iet-gtd.2018.6230.

[19] E. A. Bhuiyan, M. Z. Hossain, S. Muyeen, S. R. Fahim, S. K. Sarker a S. K. Das, "Towards next generation virtual power plant: Technology review and frameworks", *Renewable and Sustainable Energy Reviews*, roč. 150, s. 111 358, 2021, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2021.111358. URL: https://www.sciencedirect.com/science/article/pii/S1364032121006444.

[20] L. Yavuz, A. Önen, S. Muyeen a I. Kamwa, "Transformation of microgrid to virtual power plant – a comprehensive review", *IET Generation, Transmission & Distribution*, roč. 13, č. 11, s. 1994–2005, DOI: https://doi.org/10.1049/iet-gtd.2018.5649. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-gtd.2018.5649. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-gtd.2018.5649.

[21] H. Mohammadi Rouzbahani, H. Karimipour a L. Lei, "A review on virtual power plant for energy management", *Sustainable Energy Technologies and Assessments*, roč. 47, s. 101 370, říj. 2021. DOI: 10.1016/j.seta.2021.101370.

[22] G. Zhang, C. Jiang a X. Wang, "Comprehensive review on structure and operation of virtual power plant in electrical system", *IET Generation, Transmission & Distribution*, roč. 13, č. 2, s. 145–156, DOI: 10.1049/IET-GTD.2018.5880.

[23] Y. Xue, M. Starke, J. Dong et al., "On a Future for Smart Inverters with Integrated System Functions", s. 1–8, čvn. 2018. DOI: 10.1109/PEDG.2018.8447750.

[24] •, "Common Functions for Smart Inverters: 4th Edition", EPRI, Palo Alto, CA, tech. zpr., 2016.

[25] T. Morstyn, N. Farrell, S. Darby a M. Mcculloch, "Using peer-to-peer energy-trading platforms to incentivize prosumers to form federated power plants", *Nature Energy*, roč. 3, ún. 2018. DOI: 10.1038/s41560-017-0075-y.

[26] C. Zhang, J. Wu, C. Long a M. Cheng, "Review of Existing Peer-to-Peer Energy Trading Projects", *Energy Procedia*, roč. 105, s. 2563–2568, květ. 2017. DOI: 10.1016/j.egypro.2017.03.737.

[27] *The Virtual Utility: Accounting, Technology & Competitive Aspects of the Emerging Industry*. Springer New York, NY, 1997.

[28] X. Wang, Z. Liu, H. Zhang, Y. Zhao, J. Shi a H. Ding, "A Review on Virtual Power Plant Concept, Application and Challenges", s. 4328–4333, květ. 2019. DOI: 10.1109/ISGT-Asia.2019.8881433.

[29] J. Zhang, "The Concept, Project and Current Status of Virtual Power Plant: A Review", *Journal of Physics: Conference Series*, roč. 2152, č. 1, s. 012 059, 2022. DOI: 10.1088/1742-6596/2152/1/012059. URL: https://dx.doi.org/10.1088/1742-6596/2152/1/012059.

[30] •. "South Australia's Virtual Power Plant — Energy & Mining". (2022), URL: https://www.energymining.sa.gov.au/consumers/solar-and-batteries/south-australias-virtual-power-plant.

[31] •. "Virtual Power Plant in South Australia - Australian Renewable Energy Agency (ARENA)". (2022), URL: https://arena.gov.au/knowledge-bank/virtual-power-plant-in-south-australia.

[32] H. B. D. Mark Hudson Beale Martin T. Hagan. "Deep Learning Toolbox: Getting Started Guide". (2023), URL: https://www.mathworks.com/help/pdf_doc/deeplearning/nnet_gs.pdf.

[33] H. B. D. Mark Hudson Beale Martin T. Hagan. "Deep Learning Toolbox: User's Guide". (2023), URL: https://www.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf.

[34] A. Sahithi, "Simulace FV systému s ukládáním energie do vody", Bakalářská práce, České vysoké učení technické v Praze, 2022.

[35] Mathworks. "Reinforcement Learning Toolbox: User's Guide". (2023), URL: https://www.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf.

[36] Mathworks. "Train DQN Agent for Lane Keeping Assist Using Parallel Computing". (2023), URL: https://www.mathworks.com/help/reinforcement-learning/ug/train-dqn-agent-for-lane-keeping-assist-using-parallel-computing.html.

[37] H. Kazmi, J. Suykens, A. Balint a J. Driesen, "Multi-agent reinforcement learning for modeling and control of thermostatically controlled loads", *Applied Energy*, roč. 238, s. 1022–1035, 2019, ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2019.01.140. URL: https://www.sciencedirect.com/science/article/pii/S0306261919301564.