FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Diplomová práce

Použití nástrojů UI pro hledání optimálního scénáře těžby PZP

Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie

Autor práce: **Bc. Aleš Foldyna**
Vedoucí práce: doc. Ing. Otto Severýn, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání diplomové práce

Použití nástrojů UI pro hledání optimálního scénáře těžby PZP

Jméno a příjmení:	**Bc. Aleš Foldyna**
Osobní číslo:	M20000166
Studijní program:	N2612 Elektrotechnika a informatika
Studijní obor:	Informační technologie
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2021/2022

Zásady pro vypracování:

1. Seznamte se s problematikou skladování zemního plynu v geologických strukturách a existujícími metodami řízení těžby.
2. Proveďte rešerši algoritmů UI, které by bylo možno použít pro úlohu hledání optimálního těžebního scénáře.
3. Vybrané algoritmy natrénujte a otestujte na reálných provozních datech vybraného zásobníku.
4. Porovnejte výsledky algoritmů UI s výsledky numerických modelů a to z hlediska kvality výsledků a výpočetní náročnosti.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40-50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] Zangl, G., Hanner, J.: Data mining – applications in the petroleum industry. Road Oak Publishing, 2003. ISBN 0-9677248-1-8.

[2] Shi, G.: Data mining and knowledge discovery for geoscientists. Elsevier, 2014. ISBN 978-0-12-410437-2.

[3] Onderka, V., a kol.: Expertní systém PZP. Technická zpráva, RWE Brno, 2008.

Vedoucí práce:	doc. Ing. Otto Severýn, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2021
Předpokládaný termín odevzdání:	20. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2021

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

20. dubna 2023 Bc. Aleš Foldyna

POUŽITÍ NÁSTROJŮ UI PRO HLEDÁNÍ OPTIMÁLNÍHO SCÉNÁŘE TĚŽBY PZP

ABSTRAKT

Tato práce se zabývá jednou z mnoha oblastí optimalizace těžby z podzemních zásobníků plynu (PZP). Konkrétně jde o prvotní pokus vytvoření jádra expertního systému založeného na umělé neuronové síti, který by na základě celkového denního požadovaného množství těžby ze zásobníku dokázal vyhodnotit optimální rozložení celkového množství na jednotlivé sondy daného zásobníku.

Klíčová slova: Expertní systém, Umělá neuronová sít, Podzemní zásobník plynu

FINDING OF OPTIMAL UGS PRODUCTION SCENARIO BY AI TOOLS

ABSTRACT

This work deals with one of the many areas of optimization of extraction from underground gas reservoirs. Specifically, it is an initial attempt to create the core of an expert system based on an artificial neural network, which would be able to evaluate the optimal distribution to the individual probes of the given stack based on the total daily required amount.

Keywords: Expert system, Artificial neural network, Underground gas stack

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé práce panu doc. Ing. Ottovi Severýnovi, Ph. D., za vstřícnost a cenné připomínky během psaní této diplomové práce.

OBSAH

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

ANN	Artificial neural network (umělá neuronová síť)
I/O	Vstup a výstup
LSTM	Long short-term memory layer
NARX	Nelineární autoregresní neuronová síť s externím vstupem
PZP	Podzemní zásobník plynu
RMSE	Root-Mean-Square Error

1 Úvod

Operativní řízení podzemního zásobníku plynu (PZP) se každodenně zabývá několika otázkami. Jedna z nich tvoří i náplň této práce a zní: Jak nejlépe rozdělit požadované množství plynu ze zásobníku na jednotlivé produkční body (sondy)?

PZP se budují ve vhodných geologických strukturách hluboko v podzemí a jediný možný přístup k nim z povrchu země je přes tzv. sondy (hlubinný vrt a jeho vystrojení). Ty se vrtají převážně na základě nepřímých informací získaných z geologického a geofyzikálního výzkumu dané oblasti. Vzhledem k tomu, že se PZP rozkládá na rozsáhlé ploše, je v rámci jednoho zásobníku vyvrtáno většinou několik desítek sond.

Sondy mají odlišné produkční vlastnosti, a to i v rámci jednoho zásobníku. Není tedy efektivní při požadavku na odtěžení konkrétního množství plynu ze zásobníku rozdělit tuto sumu rovnoměrně mezi jednotlivé sondy. V dnešní době se v rámci řízení PZP používají simulátory založené na numerických modelech, které jsou schopné poskytnout informaci, zdali je zásobník schopen dodat požadované množství plynu pro následující těžební den. Pokud ano, simulátor rovněž navrhne rozložení na jednotlivé sondy. V praxi však často dochází k tomu, že technik navrhované rozložení ze simulátoru upravuje na základě svých vlastních zkušeností.

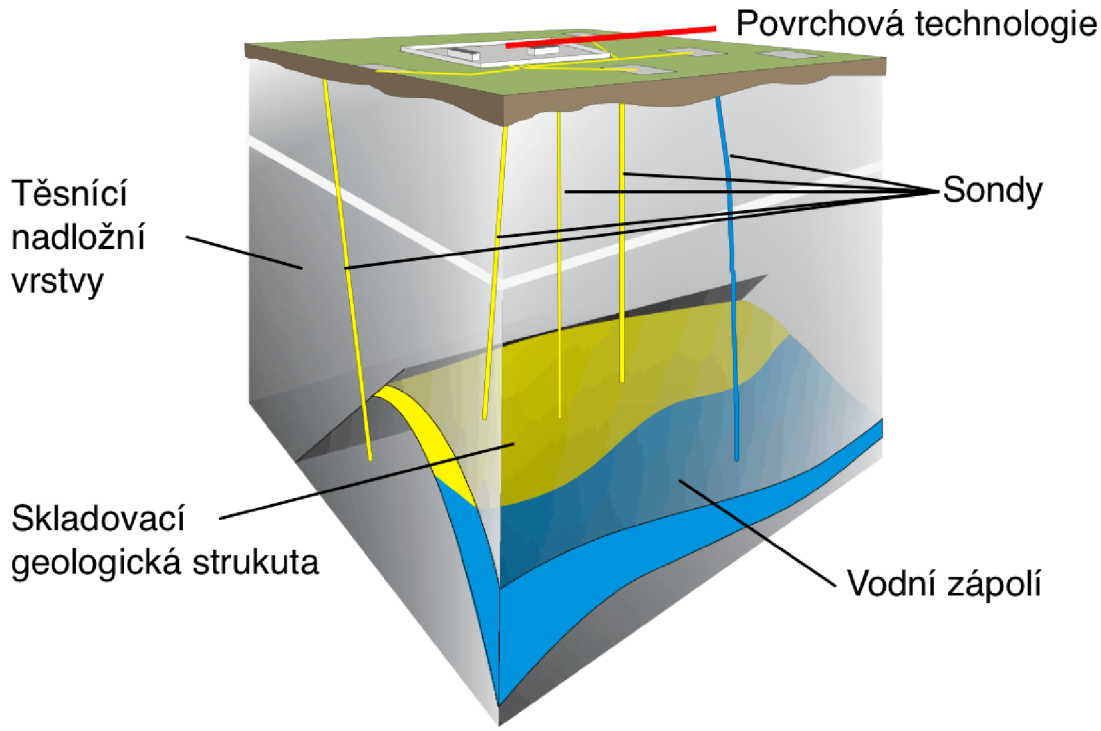
Cílem této práce je zjistit, zda by neuronová síť umožňovala lépe predikovat výše uvedené rozdělení, než predikuje dosavadní simulátor. V práci tedy dojde k natrénování vhodné neuronové sítě, která na základě dvou vstupů (požadovaného množství plynu na následující těžební den a aktuálního stavu zásob v PZP) rozhodne o optimálním rozložení celkové požadované sumy na jednotlivé sondy jednoho konkrétního zásobníku. Poté budou na testovacích datech provedeny predikce pomocí vytvořené sítě a simulátoru, které umožní porovnání obou těchto technologií.

Na začátku práce bude popsána problematika skladování zemního plynu v PZP, která umožní objasnit faktory ovlivňující rozdělení požadovaného množství plynu na jednotlivé sondy. V dalším kroku bude vybrána vhodná architektura sítě z hlediska průchodu dat sítí a bude popsáno, jak lze neuronové sítě trénovat pomocí aplikace Matlab. Poté dojde k návrhu řešení, jehož cílem bude stanovit přesné kroky, které budou v rámci práce provedeny pro dosažení cíle práce.

2 PODZEMNÍ ZÁSOBNÍKY PLYNU

Podzemní zásobník plynu [1, 2] většinou není uměle vytvořené prostředí, které by bylo přímo určené pro skladování plynu, ale jedná se o vhodnou přirozenou geologickou strukturu hluboko v podzemí, která umožňuje uchovat plyn.

Na obrázku 2.1 je znázorněno obecné schéma PZP. Geologické prostředí se skládá z těsnící nadložní vrstvy (izolátoru), která zabraňuje úniku plynu z níže položené skladovací struktury (kolektoru). Přístup povrchové technologie ke kolektoru zajišťují sondy.



Obrázek 2.1: PZP – Schéma [2]

Z výše uvedeného popisu PZP je zřejmé, že o přirozené skladovací struktuře lze získat jen velmi málo přímých informací, které tvoří jádra z vrtů a jejich následné zkoumání. Většinu informací o skladovací struktuře představují nepřímé informace (modelování z informací získaných pomocí geologic-

kých, geofyzikálních a hydrodynamických výzkumů). Další poznatky lze získat z provozních dat a následného využití numerických modelů [2].

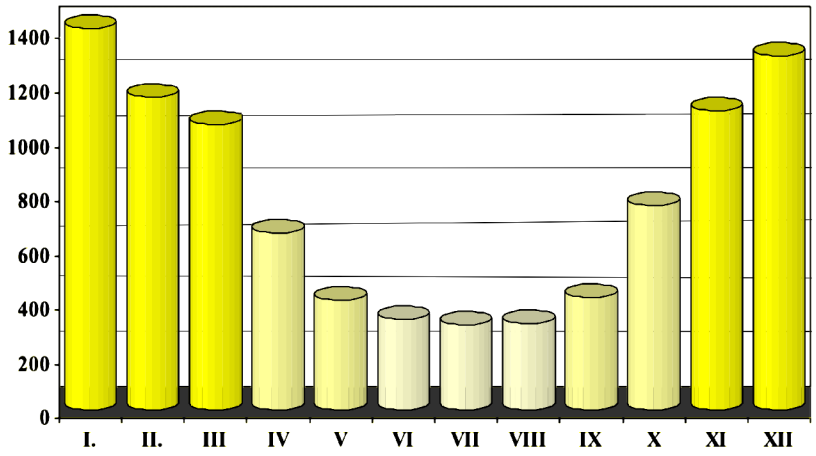
Vhodných geologických struktur je vícero druhů, jsou uvedeny a popsány v [1]. Na území ČR se z 97 % celkové skladovací kapacity využívají přirozené geologické struktury typu porézních hornin [2].

2.1 PROVOZ PZP

Zemní plyn je na území ČR využíván převážně k vytápění [1], to způsobuje velmi silnou závislost spotřeby plynu na denní teplotě, která je viditelná na obrázku 2.2.

Do roku 2022 platilo, že se na území ČR dostával plyn především z poloostrovu Jamal prostřednictvím tranzitního potrubí, kdy v letních měsících bylo možné uspokojit poptávku přímo z tranzitního potrubí a navíc z přebytku plynu bylo možné naplňovat zásobníky.

Naopak v zimních měsících, kdy spotřeba plynu výrazně stoupne, kapacita a časová prodleva proudění plynu přes tranzitní plynové potrubí neumožňovala uspokojit poptávku, docházelo pro její uspokojení k těžbě plynu z PZP.



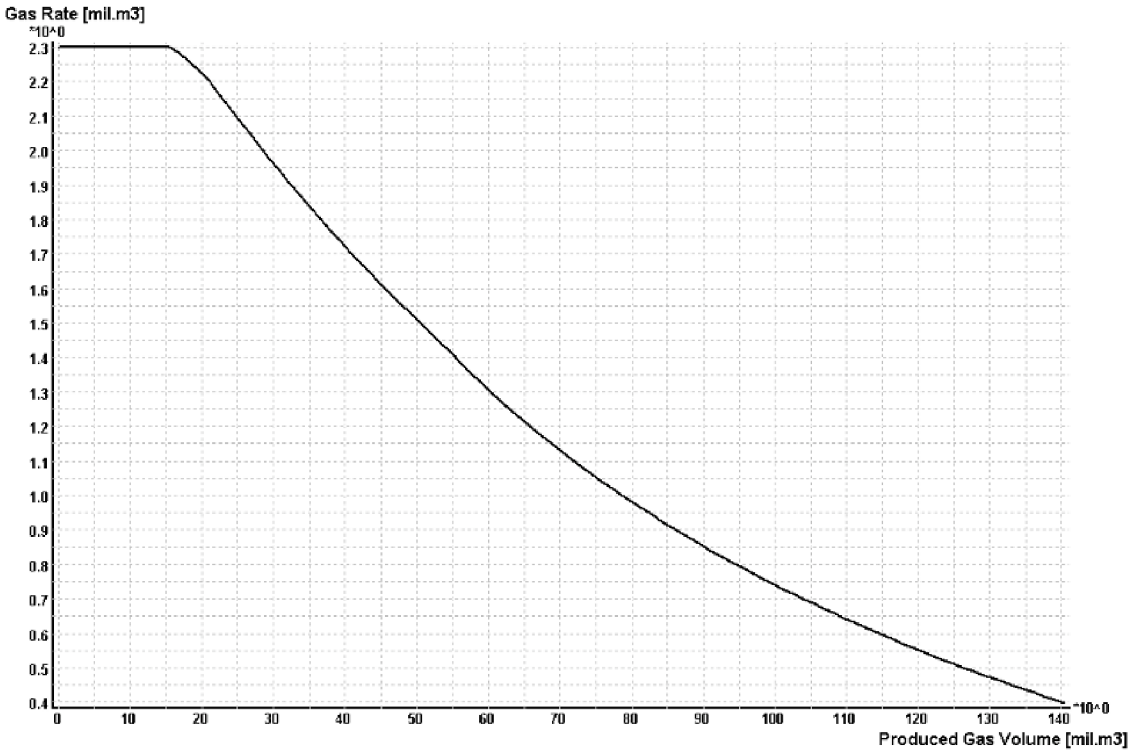
Obrázek 2.2: Průběh roční spotřeby zemního plynu v ČR [1]

2.2 ODTĚŽOVACÍ KŘIVKA

Odtěžovací křivka (viz obrázek 2.3) znázorňuje závislost maximálního denního výkonu zásobníku na stavu zásob. Dispečer na jejím základě rozhoduje, zda-li je zásobník schopen poskytnout potřebné množství plynu pro následující den [2].

Existují dva typy tvorby křivek. První typ, kdy křivka je nakreslená geologem na začátku těžby a během těžební sezóny je jednou až dvakrát aktualizována. Křivka vytvořená tímto modelem se nazývá statická a nepromítá rychlost těžby (platí pouze pro pomalou těžbu). Důvodem je omezení ložiska, které neumožňuje okamžité a rovnoměrné dorovnávání tlaků. Pokud tedy dojde k intenzivní těžbě, vzniknou kolem sond tlakové deprese, které zapříčiní snížení jejich výkonu. Naopak po odstavení zásobníku v délce dnů až týdnů se tlaky dorovnají zpět na odpovídající úroveň odtěžovací křivky [2].

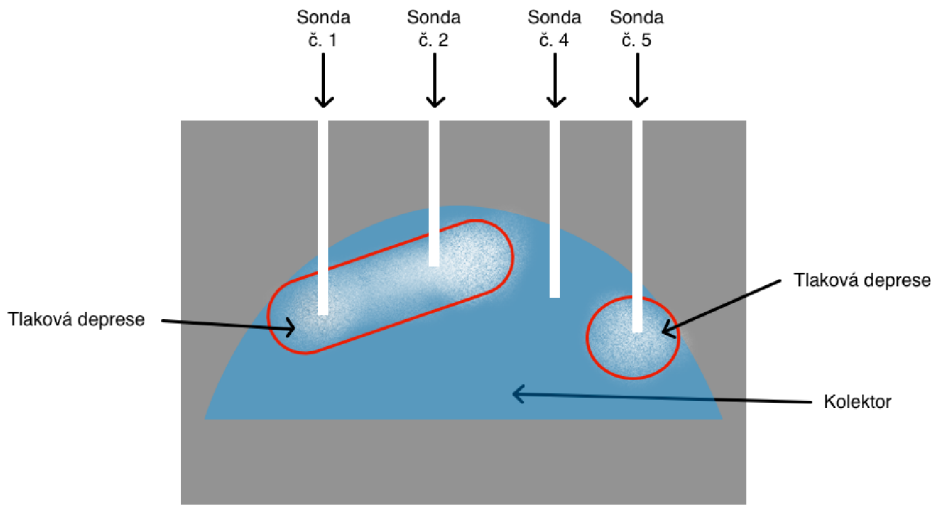
Druhým typem je automaticky generovaná odtěžovací křivka. Je založena na numerickém modelu, který se denně aktualizuje. Tím odráží skutečný stav zásobníku, kde jsou zahrnuty i údaje o poslední těžbě. Automaticky generovaná odtěžovací křivka je součástí simulátoru, který se v dnešní době používá pro řízení PZP. Simulátor kromě výše zmíněné křivky dokáže navrhnout rozložení celkového požadovaného množství plynu na jednotlivé sondy. Avšak mezi geology převládá stálá nedůvěra k navrhovanému rozložení a často dochází k jeho korekci.



Obrázek 2.3: Příklad odtěžovací křivky [2]

2.3 FAKTORY OVLIVŇUJÍCÍ TĚŽBU

V předchozí podkapitole 2.2 byl zmíněn problém vzniku tlakových depresí, které výrazně ovlivňují těžbu z PZP. Na obrázku 2.4 je zobrazeno schéma PZP, kde modrá barva znázorňuje tlak plynu dle odtěžovací křivky. Pokud dojde k intenzivní těžbě, vzniknou kolem sond, které se podílely na těžbě, tlakové deprese (znázorněné na obrázku bílou barvou v červeném ohraničení). Z toho vyplývá, že při rozložení požadovaného množství plynu na jednotlivé sondy se musí zohlednit i předešlá těžba.



Obrázek 2.4: Ukázka „depresní kapsy“

Jelikož se místo a hloubka sondy určuje převážně na základě nepřímých informací, nelze sondy umístit tak, aby byly předem známé přesné vlastnosti porézní horniny v umístění sondy, která má vliv na její výkon. To znamená, že i v rámci jednoho zásobníku se výkonnost každé sondy výrazně liší.

Dalším faktorem, který ovlivňuje těžbu, je tzv. interference sond. Ta vzniká při větší propustnosti kolektoru mezi blízkými sondami. Větší propustnost zapříčiní, že se tlaková deprese jedné sondy projeví na výkonnosti druhé sondy (znázorněno na obrázku 2.4 u sond č. 1 a 2).

Výše uvedené faktory poskytují tři klíčové informace pro účely této práce:

I. Každá sonda má jiné výkonové parametry (maximální množství vytěženého plynu za den).

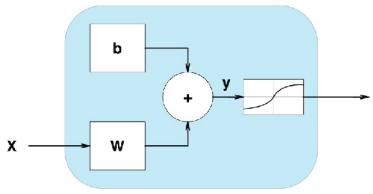
II. Sondy se mohou navzájem ovlivňovat.

III. Historie těžby dané sondy má vliv na její výkon při další těžbě.

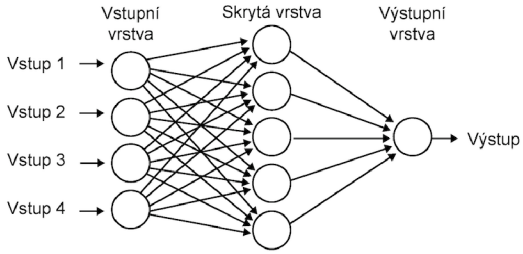
3 NEURONOVÉ SÍTĚ

Pod pojmem umělá neuronová síť (ANN, z anglického názvu Artificial neuron network) se myslí algoritmus, který je inspirovaný modely zpracování smyslových vjemů mozkem. Použitím algoritmů, které napodobují procesy skutečných neuronů, můžeme síť přimět, aby se „naučila“ řešit mnoho typů problémů [3].

Základní stavební jednotkou neuronové sítě je umělý neuron. Umělé neurony se vrství a propojují a tím vzniká neuronová síť. Neurony mohou mít různou podobu, jedna z nich je znázorněna na obrázku 3.1. Jde v podstatě o rovnici přímky $y = x \cdot w + b$, kde výstup projde ještě tzv. aktivační funkcí. Ta se požívá pro zabránění linearity. Kdyby síť neobsahovala aktivační funkce, jednalo by se pouze o složenou funkci více lineárních funkcí a konečný výsledek by byl vždy také lineární [21]. Jako aktivační funkce se často používá například Hyperbolická tangenta, Sigmoida, reLU, atd.



Obrázek 3.1: Umělý neuron



Obrázek 3.2: Schéma neuronové sítě [20]

Pomocí algoritmu zpětné propagace se u neuronů iteračně nastavují váhy w a bias b (učící se parametry) tak, aby se minimalizoval rozdíl mezi výstupem sítě a chtěnými výstupy (předlohami). Rozdíl mezi výstupem sítě a předlohami se používá pro výpočet gradientu, který se posléze propaguje algoritmem zpětného průchodu. Nastavením učících se parametrů je pak neuron schopen řešit lineární problém.

Zapojení více umělých neuronů a jejich vzájemné propojení umožňuje neuronové síti řešit pak i nelineární problémy různých typů.

3.1 DEFINICE PROBLÉMU PRÁCE

Cílem této práce je vytvoření neuronové sítě, která bude na základě dvou vstupních informací:

x_1 – požadovaného množství plynu pro následující těžební den

x_2 – aktuálního stavu zásob v PZP

generovat N výstupních hodnot, kde N označuje počet sond konkrétního zásobníku. Výstupní hodnoty by pak měly reprezentovat vhodné rozložení vstupu x_1 tak, aby těžba z PZP byla efektivní.

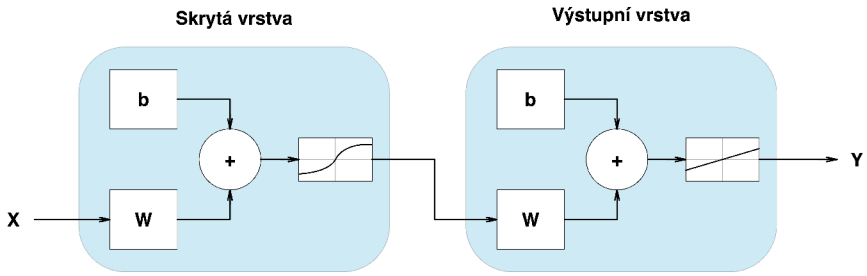
Jedná se tedy o úlohu typu regrese, přičemž z informací uvedených v kapitole 2 lze vyvodit, že historie těžby z jednotlivých sond má vliv na budoucí těžbu.

3.2 POTENCIÁLNĚ VHODNÉ TYPY SÍTÍ

Umělé neuronové sítě je možné rozlišovat podle velkého počtu kritérií (průchodu dat sítí, použitými typy neuronů, atd.). V této části práce budou popsány vybrané pojmenované typy sítí a zhodnotí se jejich potenciální vhodnost pro výše uvedený problém.

3.2.1 Dopředná síť

Dopřednou sítí se myslí neuronová síť, kterou prochází pouze aktuální vstupní data [4]. Na obrázku 3.3 je znázorněné zjednodušené schéma dopředné sítě, kde neuron z obrázku 3.1 reprezentuje perceptron (několik neuronů v jedné vrstvě). Výstupní vrstva se přidává do sítí z důvodu transformace dat na chtěný počet výstupů (v rámci řešeného problému bude mít tato vrstva vždy N neuronů).

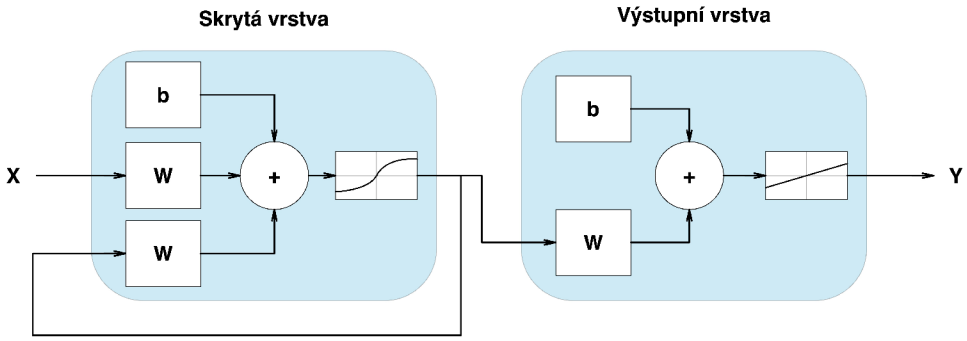


Obrázek 3.3: Schéma dopředné sítě

Vzhledem k tomu, že tato síť neumožňuje zohlednit předchozí výstupy, které poskytují důležité informace v definovaném problému, nelze tuto síť považovat za vhodnou k účelům této práce.

3.2.2 Rekurentní síť

Rekurentní síť je architektura, která umožňuje poskytnout zpětnou vazbu z předchozího rozhodování sítě prostřednictvím rekurentní vrstvy [5]. Ze schématu sítě na obrázku 3.4 je vidět, že každý neuron ve skryté vrstvě použije svůj historický výstup jako další vstup k aktuálnímu vstupu.

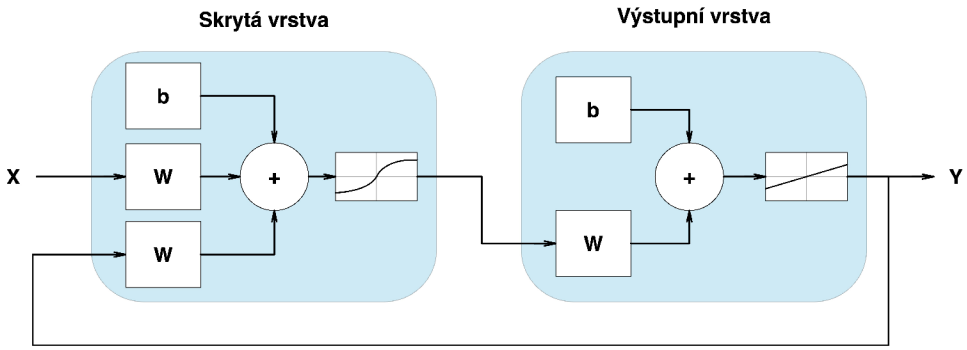


Obrázek 3.4: Schéma rekurentní sítě

Tato architektura se používá pro časové řady, kterým odpovídá i úloha této práce. Proto ji lze považovat za potenciálně vhodnou pro řešení daného problému.

3.2.3 NARX

Nelineární autoregresní neuronová síť s externím vstupem (dále jen NARX) je architektura znázorněna na obrázku 3.5, která je schopná se naučit předpovídat jednu časovou řadu na základě téže časové řady ze zpětnovazebního vstupu a vnější (exogenní) časové řady [6].



Obrázek 3.5: Schéma NARX

Z informací uvedených v kapitole 2 si lze představit vytěžené množství plynu konkrétní sondy konkrétního zásobníku jako jednu časovou řadu s periodou t, kde t odpovídá jednomu dni. Dále celkové požadované množství jako první exogenní vstup a aktuální stav zásob jako druhý exogenní vstup. Pak je síť této architektury schopna se naučit všechny vlastnosti uvedené v kapitole 2.3 a je tedy vhodná pro problém této práce.

4 TVORBA NEURONOVÝCH SÍTÍ POMOCÍ MATLABU

Rozšíření aplikace Matlab nástrojem pro hluboké učení neuronových sítí „Deep learning toolbox“ umožňuje prostřednictvím této aplikace efektivně skládat a učit neuronové sítě [7]. Tento nástroj přidá do aplikace Matlab dvě různé architektury pro tvorbu a učení neuronové sítě, které jsou na sobě nezávislé a nelze je navzájem kombinovat.

První z nich je určena pro mělké neuronové sítě (síť obsahuje jednu skrytou vrstvu, nebo pouze malé množství skrytých vrstev)[8] a umožňuje velmi rychlou a snadnou kompletní implementaci trénování a testování mělkých sítí včetně pre-processingu dat, ale neumožňuje výraznější flexibilitu v definování vlastních vrstev, vytvoření vlastní tréninkové smyčky, atd.

Druhá architektura naopak slouží k vytváření hlubokých neuronových sítí (síť obsahuje větší počet skrytých vrstev) [9] a umožňuje v podstatě neomezenou flexibilitu za předpokladu rozsáhlejších znalostí z oblasti neuronových sítí.

V této části práce budou uvedeny k oběma výše zmíněným architekturám základní informace, které umožní vytvořit a natrénovat neuronovou síť pro předmětný problém za pomocí obou architektur. Níže uvedené informace jsou platné pro verzi Matlabu R2022b a verzi „Deep learning toolbox“ 14.5; tyto verze byly aktuální v době vzniku této práce.

4.1 MĚLKÉ NEURONOVÉ SÍTĚ

Architektura pro vytvoření mělké neuronové sítě je zastřešena jednou třídou, nazvanou „network“. Konfigurací tohoto objektu lze vytvořit síť s M skrytými vrstvami, kde každá z nich může obsahovat N neuronů. Podrobné informace o objektu lze získat z dokumentace [10].

Pro zjednodušení vytvoření sítě pomocí objektu „network“ je možné využít několik funkcí, které připraví objekt tak, aby odpovídal potřebné architektuře sítě. Pro architektury uvedené v kapitole 3 jsou názvy funkcí odpovídající dané architektuře znázorněny v tabulce 4.1.

Architektura sítě	Funkce
Dopředná síť	fitnet
Rekurentní síť	layrecnet
NARX	narxnet

Tabulka 4.1: Funkce pro vytvoření neuronové sítě

4.1.1 Data

Data pro trénování sítě se předávají až do trénovací funkce popsané v kapitole 4.1.3 ve dvou možných datových typech, jimiž jsou matice a buňky. Ukázka kódu 4.1 znázorňuje příklad obou možných validních formátů dat pro trénování sítě, která má dva vstupy a pět výstupů. Ve formátu matice řádky představují jednotlivé vstupy do sítě X (případně předlohy T) a sloupce pak jednotlivé záznamy (v ukázce kódu 4.1 má příklad 100 záznamů). Pro formát buněk je řádek jenom jeden a počet sloupců zůstává roven počtu jednotlivých záznamů, přičemž každá buňka obsahuje všechny vstupy nebo předlohy konkrétního záznamu pro trénování sítě.

```
1  X = rand(2, 100); % Vstupy x Záznamy
2  T = rand(5, 100); % Predlohy x Záznamy
3
4  cellX = mat2cell(X, size(X, 2), ones(1, size(X, 1)));
5  cellT = mat2cell(T, size(T, 2), ones(1, size(T, 1)));
```

Ukázka kódu 4.1: Validní formáty trénovacích dat pro funkci train

Rozložení dat na trénovací, validační a testovací sadu se provádí nastavením proměnných „divideFcn“ a „divideParam“ v objektu „network“. Matlab poskytuje čtyři funkce (viz tabulka 4.2), mezi nimiž lze volit způsob rozdělení dat. Prostřednictvím první z uvedených proměnných „divideFcn“ se volí, která z funkcí se má použít zadáním názvu funkce jako pole znaků.

Rozdělení	Funkce
Náhodné	dividerand
Po souvislých blocích	divideblock
Pomocí prokládaného výběru	divideint
Podle přesně určených indexů	divideind

Tabulka 4.2: Funkce pro rozložení dat

Proměnná „divideParam“ je typu struktury a slouží k zadání poměrů rozložení dat na jednotlivé sady. Při zvolení funkce „divideind“ se rozdělení zadává výčtem indexů, u ostatních funkcí procentuálním rozložením.

Pro sítě s časovým posunem je v prostředí Matlab implementována funkce „preparets“, která na základě objektu „network“, vstupních a předlohových dat připraví data tak, aby se dala přímo předat do trénovací funkce.

4.1.2 Standardizace

Standardizace převádí jednotlivé proměnné na stejné měřítko, přestává tedy záležet na skutečném rozměru příslušných proměnných [20]. Standardizaci vstupů i výstupů sítě je možné provést přímo pomocí objektu „network“, ve kterém proměnné „inputs“ a „outputs“ uchovávají informace o vstupech a výstupech sítě. Tyto proměnné jsou formátu buněk, přičemž buňky obsahují na příslušné pozici vstupů a výstupů sítě strukturu, v níž lze mimo jiné nastavit standardizaci pomocí proměnné „processFcns“. Standardizačních funkcí lze pro konkrétní I/O nastavit více, jak je to znázorněno v ukázce kódu 4.2 na prvním řádku. Implementovatelné standardizační funkce, které lze využít, jsou uvedeny v tabulce 4.3. Pro případ využití neimplementované standardizační metody se může proměnná „processFcns“ nechat prázdná a standardizace může být provedena mimo síť.

Standardizační funkce	**Funkcionalita**
mapminmax	Transformuje I/O do rozsahu $\langle -1, 1 \rangle$.
mapstd	Převede I/O tak, aby měly nulovou střední hodnotu a jednotný rozptyl.
processpca	Zpracuje sloupce I/O pomocí analýzy hlavních komponent.
fixunknowns	Nahradí NaN hodnoty průměrem řádku.
removeconstantrows	Odstraní konstantní hodnoty I/O.

Tabulka 4.3: Implementované standardizační funkce

```
1  net.inputs{1}.processFcns={'removeconstantrows','mapminmax'};
2  net.outputs{2}.processFcns={'mapminmax'};
```

Ukázka kódu 4.2: Definice standardizačních funkcí objektu „network“

4.1.3 Trénování

Před spuštěním trénování sítě je potřeba v objektu „network“ nastavit trénovací algoritmus a jeho parametry pro trénování. Uživatel má na výběr z dvanácti známých algoritmů, jejichž výčet, funkcionalita a potřebné parametry jsou uvedeny v dokumentaci [11]. Při vytvoření objektu „network“ pomocí

funkcí uvedených v tabulce 4.1 se jako výchozí metoda použije algoritmus Levenberg-Marquardt zpětné propagace s výchozími parametry.

Trénování neuronové sítě vytvořené architekturou pro tvorbu mělkých sítí se vyvolá funkcí „train“ [12], do které se předá objekt „network“, vstupní hodnoty a jejich předlohy (pro sítě s časovým posunem lze využít výstup z funkce „preparets“ zmíněné v podkapitole 4.1.1). Tímto krokem se spustí trénovací proces s grafickým rozhraním (znázorněné na obrázku 4.1), který se ukončí po dovršení jedné z podmínek pro ukončení konkrétního trénovacího algoritmu, a funkce vrátí objekt typu „network“ s již natrénovanými parametry.

Obrázek 4.1: Grafické rozhraní zobrazené funkcí „train“

Výše zmíněné grafické rozhraní poskytne veškeré potřebné informace včetně výsledků jednotlivých datových sad pro jednotlivé iterace trénování.

4.1.4 Testování

Jak již bylo zmíněno v předchozí podkapitole, trénovací funkce provede i dopředný průchod na testovacích datech a výsledky zobrazí v grafické podobě.

Ukázka kódu 4.3 znázorňuje dva možné způsoby, jak simulovat dopředný průchod sítí. Pro simulaci sítě slouží funkce „sim“, která je implementována v objektu sítě, ale lze ji použít i samostatně. Simulace prostřednictvím implementace v objektu „network“ se provádí použitím natrénované sítě jakožto funkce, které se předají vstupní hodnoty, případně vnitřní stavy z předchozích průchodů pro časově zpožděné sítě.

```
1  y = sim(net,x); % simulace pomocí funkce sim
2  y = net(x); % vyvolání funkce sim pomocí objektu network
```

Ukázka kódu 4.3: Simulace sítě

4.2 HLUBOKÉ NEURONOVÉ SÍTĚ

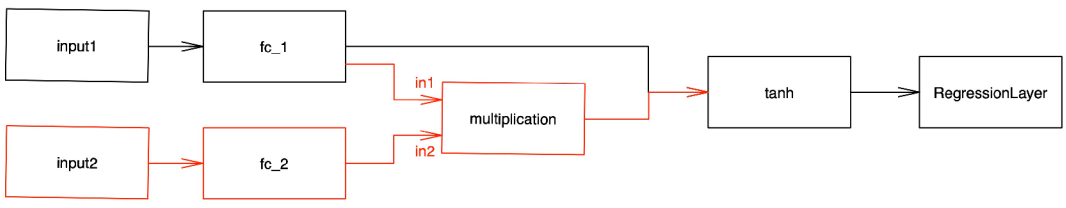
Druhá z architektur pro učení neuronových sítí pomocí nástroje „Deep learning toolbox“ aplikace Matlab je zcela odlišná od architektury popsané v podkapitole 4.1. Je založena na třídách jednotlivých vrstev dědících ze tříd v balíčku „nnet.layer“. Síť se pak skládá pomocí těchto tříd způsobem, který bude podrobněji popsán v podkapitole 4.2.1. Tato architektura vytváření sítí umožňuje sestavení prakticky libovolné umělé neuronové sítě.

4.2.1 Vytvoření sítě

Základem pro architekturu „Deep learning toolboxu“ jsou objekty jednotlivých vrstev, které se skládají do pole, a tím vzniká neuronová síť. V rámci nástroje je připraveno velké množství tříd pro různé známé problémy; jejich výčet je uveden v dokumentaci [13]. Pro případ, kdy by nebyla vhodná vrstva implementována, lze definovat vlastní vrstvu, přičemž postup pro její vytvoření je uveden v dokumentaci [14]. Třídy vrstev je možné rozdělit do tří kategorií podle jejich postavení v síti na vstupní vrstvy, mezivrstvy a výstupní vrstvy. Poslední zmiňovaná vrstva není ve smyslu uvedeném v podkapitole 3.2.1, ale jedná se o vrstvu, pomocí které se vypočítává v rámci procesu učení gradient.

Síť je možné vytvořit dvěma způsoby. Pro jednoduché sítě, které neobsahují větvení dat při průchodu sítí, lze použít pole obsahující jednotlivé objekty vrstev v hierarchickém pořadí průchodu dat sítí. Pro větvené sítě je nutné použít objekt „layerGraph“, který umožní libovolně namapovat jednotlivé průchody sítí.

Na obrázku 4.2 jsou znázorněné dvě varianty sítí. První síť, která se skládá pouze z černých prvků a je tedy tvořena právě jednou větví průchodu dat sítí, lze vytvořit jako pole obsahující jednotlivé objekty vrstev (viz ukázka kódu 4.4). Pro druhou síť, která je složena z první sítě přidáním červených vrstev, se již musí použít objekt „layerGraph“. Pomocí funkce „addLayers“ jsou do objektu postupně předány jednotlivé celistvé sekvence vrstev, které jsou tvořeny hierarchickým průchodem dat, a následně se pomocí funkce „connectLayers“ tyto sekvence dle potřeby propojí (viz ukázka kódu 4.5).



Obrázek 4.2: Schéma sítě

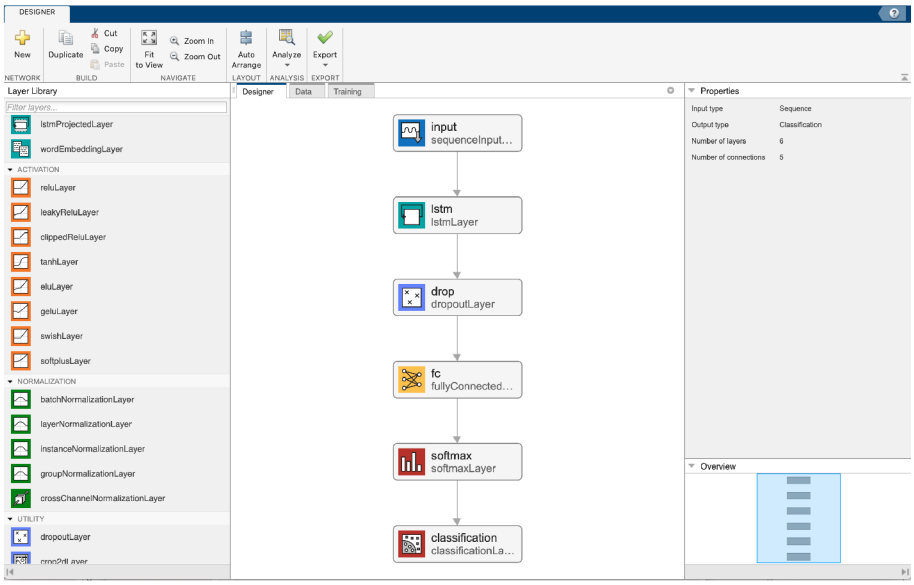
```
layers = [
    featureInputLayer(1,"Name","input1")
    fullyConnectedLayer(10,"Name","fc_1")
    tanhLayer("Name","tanh")
    regressionLayer("Name","regressionLayer")];
```

Ukázka kódu 4.4: Vytvoření sítě pomocí pole

```
lgraph = layerGraph();
tempLayers = [
    featureInputLayer(1,"Name","input1")
    fullyConnectedLayer(10,"Name","fc_1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    featureInputLayer(1,"Name","input2")
    fullyConnectedLayer(10,"Name","fc_2")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    multiplicationLayer(2,"Name","multiplication")
    tanhLayer("Name","tanh")
    regressionLayer("Name","regressionLayer")];
lgraph = addLayers(lgraph,tempLayers);

lgraph = connectLayers(lgraph,"fc_1","multiplication/in1");
lgraph = connectLayers(lgraph,"fc_2","multiplication/in2");
```

Ukázka kódu 4.5: Síť vytvořená pomocí objektu layerGraph

Pro snazší vytvoření sítě prostřednictvím architektury pro hluboké neuronové sítě je možné využít nástroj „deepNetworkDesigner“, který umožňuje vytvářet sítě v grafickém prostředí a následně kódy generovat automaticky.



Obrázek 4.3: Ukázka nástroje „deepNetworkDesigner“

4.2.2 Data

Data pro trénování lze předat trénovací funkci ve více datových typech závislých na povaze dat. Existuje ovšem jeden univerzální pro všechny možné typy dat, a to je prostřednictvím datového úložiště „Datastore“.

Uvažujme příklad neuronové sítě s dvěma vstupy a pěti výstupy, kterou chceme natrénovat pomocí trénovací sady se stem záznamů. Vstupní data a předlohy máme v datových typech matice, kde řádky obsahují jednotlivé vstupy/výstupy a sloupce záznamy. Pak ukázka kódu 4.6 znázorňuje postup, jak tato data připravit do datového úložiště. Pomocí funkce „arrayDatastore“, které se předá matice a určí se správná dimenze obsahující jednotlivé záznamy, se vytvoří objekt „ArrayDatastore“. Pro kombinaci vstupních a předlohových datových úložišť se využije funkce „combine“, která vrátí objekt „CombinedDatastore“, jenž je možný předat trénovací funkci.

```
X; % matrix (2, 100)
T; % matrix (5, 100)

adsX = arrayDatastore(X, IterationDimension=2);
adsT = arrayDatastore(T, IterationDimension=2);

cdsTrain = combine(adsX, adsT);
```

Ukázka kódu 4.6: Data pro trénování sítě

V předchozí architektuře pro vytváření mělkých sítí popsané v podkapitole 4.1 se definovalo rozdělení dat na trénovací, validační a testovací sadu v rámci zastřešujícího objektu „network“. V této architektuře pro hluboké neuronové sítě takováto možnost není k dispozici a je třeba data rozdělit ručně. Trénovací data se pak předávají přímo do trénovací funkce, validační data do nastavení parametrů pro trénování (viz níže v kapitole 4.2.3) a testovací data se použijí až při testování natrénované sítě (viz kapitola 4.2.4).

Architektura pro hluboké neuronové sítě nenabízí možnost pre-processingu a post-processingu dat. Data je tedy nutné standardizovat mimo síť.

4.2.3 Trénování

Trénování sítě vytvořené polem obsahujícím jednotlivé vrstvy, nebo objektem „layerGraph“ se vyvolá pomocí funkce „trainNetwork“ [15]. Té se předají trénovací data, vytvořená síť a nastavení parametrů pro trénování.

Funkce „trainingOptions“ vytvoří jeden ze tří možných objektů, které slouží jako nastavení parametrů trénování sítě pro funkci „trainNetwork“. Objekty jsou závislé na zvolené metodě optimalizace úpravy vah sítě vzhledem ke gradientům, přičemž architektura nástroje pro hluboké sítě nabízí právě tři metody optimalizace, jejichž podrobnější popis lze najít v dokumentaci [16].

1. Stochastický gradientní sestup s hybností (SGDM).
2. Optimalizátor RMSProp.
3. Optimalizátor Adam.

Funkce vytvářející objekt pro nastavení trénovacích parametrů „trainingOptions“ má pouze jeden povinný parametr, a to zvolení jedné z výše uvedených metod řešení optimalizace úpravy vah. Dále se do funkce mohou pomocí pojmenovaných parametrů specifikovat jednotlivá nastavení trénovacího procesu. V tabulce 4.4 jsou uvedeny parametry týkající se minidávek (počet záznamů, které vstoupí hromadně do jedné iterace trénování) a validace. Výčet všech parametrů a jejich výchozí hodnoty včetně nastavitelných hodnot pro jednotlivé metody úpravy vah jsou uvedeny a popsány v dokumentaci [16].

Parametr	Popis
MaxEpochs	Nastavení maximálního počtu epoch trénování.
MiniBatchSize	Nastavení velikosti dávky.
Shuffle	Promíchání trénovacích dat, kde lze vybrat ze tří variant: jednou na začátku trénování, před každou epochou, nebo nikdy.
ValidationData	Předání validační datové sady.
ValidationFrequency	Frekvence v iteracích provádění průchodu na validačních datech.
ValidationPatience	Maximální trpělivost validace (proces trénování se ukončí, pokud se výsledky validačních dat nebudou po určitou dobu zlepšovat).
OutputNetwork	Zvolení výstupní sítě, kde lze vybrat síť z poslední iterace, nebo síť s nejlepším validačním výsledkem.

Tabulka 4.4: Vybrané pojmenované parametry funkce „trainingOptions“

Pro případy, kdy z nějakého důvodu funkce „trainNetwork“ nevyhovuje potřebě trénování neuronové sítě, lze definovat vlastní trénovací smyčku. Neuronová síť pak musí být zastřešena objektem „dlnetwork“ a data přetypována na speciální typ „dlarray“. Síť v takovém případě nesmí obsahovat výstupní vrstvu, protože ztráta a zpětná propagace se počítá a programuje v rámci trénovací smyčky. Více informací o trénovaní neuronových sítí pomocí vlastní smyčky lze najít v dokumentaci [17, 18, 19].

4.2.4 Testování

Simulace sítě se provádí pomocí dvou funkcí, jejichž výběr závisí na typu úlohy sítě. Pro úlohu klasifikace je k dispozici funkce „classify“ a pro regresní úlohy funkce „predict“. Oběma funkcím se předá objekt natrénované sítě a testovací vstupní data.

Pro sítě obsahující vrstvy s vnitřními stavy lze simulovat i průběžné výsledky jednotlivých sekvencí prostřednictvím funkcí „predictAndUpdateState“ a „classifyAndUpdateState“, které vracejí objekt sítě se změněnými vnitřními stavy po průchodu jednotlivých dat sekvence a výstupy sítě. Pro obnovení vnitřních stavů je k dispozici funkce „resetState“.

5 NÁVRH ŘEŠENÍ PRÁCE

Řešení úlohy pomocí neuronových sítí s sebou přináší mnoho možností, jak sestavit danou síť, a rovněž velké množství různých konfigurací trénování sítě. V této části bude popsán přesný postup řešení práce.

5.1 POSTUP ŘEŠENÍ

Níže jsou uvedené jednotlivé kroky, které budou v rámci této práce provedeny s cílem natrénování neuronové sítě, která poté bude schopna řešit daný problém. Po jejich vypracování bude možné tuto práci označit za dokončenou.

1. Příprava dat:
   (a) Analýza dat
      - Zkoumání závislostí vstupu na výstupu
      - Zvážení použití transformační metody jednotlivých vstupů a výstupů
      - Výběr standardizační metody
   (b) Rozdělení dat na trénovací, validační a testovací sadu
2. Natrénování sítě typu NARX pomocí architektury pro mělké sítě popsané v kapitole 4.1, přičemž dojde k optimalizaci parametrů:
   (a) Počet neuronů ve skryté vrstvě
   (b) Zpoždění zpětné vazby
3. Návrh alespoň dvou řešení, která by potenciálně mohla vést ke zlepšení nejlepšího výsledku dosaženého v rámci bodu 2 postupu řešení práce.
4. Aplikace návrhů z bodu 3 pomocí architektury pro hluboké neuronové sítě popsané v kapitole 4.2.
5. Porovnání sítě s nejlepší výkonností dosaženou v rámci této práce s výsledky simulátoru, který se v dnešní době používá pro řízení PZP, a to z hlediska dosažených výsledků a složitosti algoritmu.

5.2 VÝKONNOST SÍTĚ

Pro stanovení výkonnosti sítě se v rámci této práce využije hodnot střední kvadratické chyby (RMSE z anglického Root-Mean-Square Error) aplikované na testovací sadě dat. RMSE je definovaná vzorcem (5.1), kde značí:

n počet záznamů testovacích dat

m počet sond

Y výstupy sítě

T požadované výsledky (předlohy)

$$RMSE = \sqrt{\frac{1}{n \cdot m} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} (Y_{ij} - T_{ij})^2 \right)} \tag{5.1}$$

6 PŘÍPRAVA DAT

V této části práce došlo k přípravě a následnému analyzování poskytnutých dat. Data bylo nutné nejdříve upravit tak, aby reprezentovala vstupy do sítě a k nim příslušné předlohy (chtěné výstupy ze sítě). Následně z nich byla vybrána pouze data, která odpovídala těžbě. Nakonec byla vybraná data analyzována.

6.1 POSKYTNUTÁ DATA

Data pro účely této práce byla poskytnuta prostřednictvím souboru typu „xlsx“, který tvoří přílohu 1a. Tabulka obsahuje uspořádaná provozní data jednoho PZP, a to po jednotlivých dnech od 1. 11. 2004 do 19. 11. 2009, kde jednotlivé sloupce obsahují informace uvedené v tabulce 6.1.

Sloupec	Poskytnutá informace
A	Datum pro daný záznam.
B	Požadované množství plynu pro těžbu z / vtlačení do PZP daného dne (hodnoty jsou uvedeny v sm^3 se záporným znaménkem pro těžbu a kladným pro vtláčení).
C–G	Rozdělení požadovaného množství na jednotlivé sondy (záporné hodnoty pro těžbu, kladné pro vtláčení).
H	Zásoby plynu v PZP po daném dni.

Tabulka 6.1: Poskytnuté informace v jednotlivých sloupcích

Vzhledem k tomu, že pro práci bylo potřeba získat pro daný den informaci o stavu zásob před denním požadavkem, byl výše uvedený soubor upraven tak, aby sloupec H obsahoval zásoby plynu v PZP před těžbou/vtlačením daném dni (příloha 1b). Následně byla takto upravená data převedena do formátu „TimeTable“ aplikace Matlab (skript 2(c)i) a uložena pro další užití v práci do souboru typu „MAT“ (příloha 1(d)i) .

6.2 VÝBĚR TĚŽEBNÍCH DAT

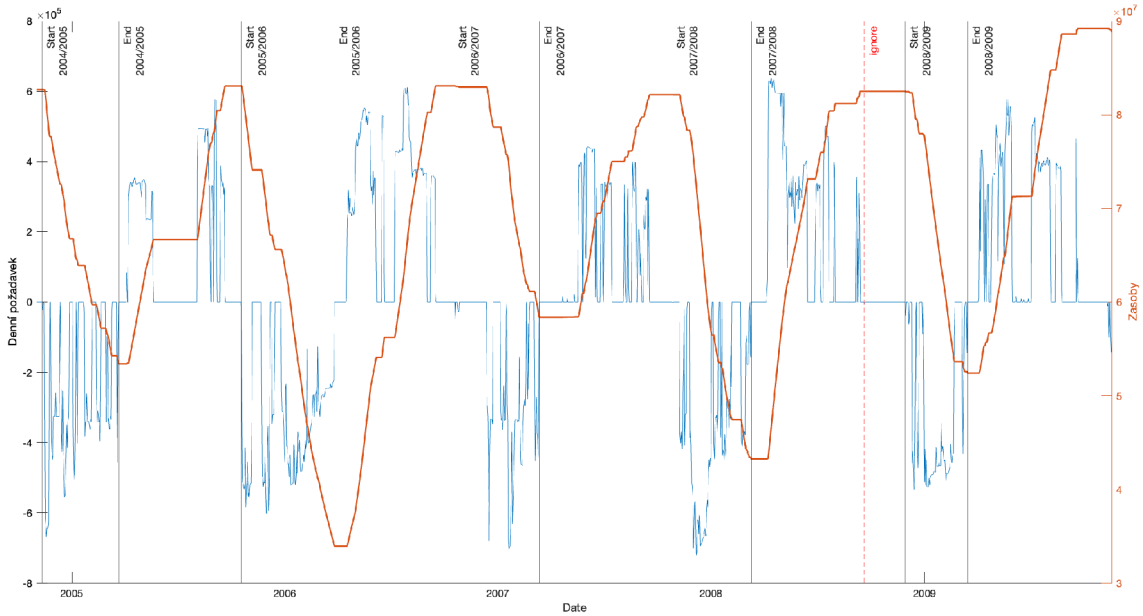
Poskytnutá data obsahují kompletní záznamy po jednotlivých dnech za období uvedené v podkapitole 6.1, tedy včetně záznamů o vtláčení do zásobníku. Vzhledem k tomu, že se tato práce má věnovat optimalizaci těžby z PZP, nikoliv vtláčení, byla z poskytnutého setu vybrána data týkající se pouze těžby.

Na základě informací uvedených v podkapitole 2.3 lze konstatovat, že nelze vybrat pouze záznamy, ve kterých došlo k těžbě, ale je nutné brát v potaz i dny, ve kterých k těžbě nedošlo, ale jejich okolí obsahuje těžební dny. Dále pak z poznatků uvedených v podkapitole 2.1 a následně i z poskytnutých dat je zřejmé, že těžba a vtláčení probíhají střídavě po určitém časovém období. Na základě těchto informací byly v práci vybrány celé tyto úseky odpovídající těžbě (dále jen „sezóny“).

Obrázek 6.1 znázorňuje v grafické podobě poskytnuté hodnoty reprezentující oba vstupy do chtěné neuronové sítě v průběhu času. Dále jsou na obrázku znázorněny začátky a konce jednotlivých těžebních sezón, které byly vybrány následujícím postupem:

1. Za začátek těžební sezóny se považuje den, kdy došlo k těžbě a předešlý nenulový záznam obsahuje vtláčení, nebo začátek datového setu.

2. Za konec těžební sezóny se považuje den, ve kterém došlo k těžbě a následný nenulový záznam obsahuje vtláčení.



Obrázek 6.1: Zásoby v PZP a denní požadavky v průběhu času

Během výběru byl ignorován záznam ze dne 20. 9. 2008 (na obrázku 6.1 znázorněn červenou svislou čárou), který by dle bodu 1 výše uvedeného postupu měl být považován za začátek těžební sezóny 2008/2009, ale vykazoval známky nestandardnosti vzhledem k ostatním záznamům.

Výše popsaný výběr dat byl proveden skriptem 2(c)ii, kde data byla vzhledem k malému počtu sezón vybrána ručně. Sezóny byly uloženy do proměnné „season“ typu struktury s dvěma proměnnými „begin“ a „end“, kde obě uchovávají pole s příslušnými dny ve formátu „datatime“. Následně byla tato proměnná „season“ uložena do souboru „MAT“ (příloha 1(d)ii).

Pro usnadnění dalšího použití dat v práci byl vytvořen objekt 2(b)i, který na základě dat z 1(d)i, 1(d)ii a rozdělení jednotlivých sezón na trénovací, validační a testovací sadu poskytuje pomocí veřejných funkcí jednoduchá pole s potřebnými daty. K objektu byla vytvořena dokumentace a seznam funkcí s jejich funkcionalitou je dostupný příkazem „help“.

6.3 Rozdělení dat

Rozdělení dat na trénovací, validační a testovací sady bylo provedeno po jednotlivých sezónách, přičemž tři sezóny byly určeny pro trénování a po jedné sezóně bylo určeno na validaci a testování. Konkrétní rozdělení je uvedeno v tabulce 6.2.

Sezóna	2004/2005	2005/2006	2006/2007	2007/2008	2008/2009
Sada	Trénovací	Trénovací	Trénovací	Validační	Testovací

Tabulka 6.2: Rozdělení dat

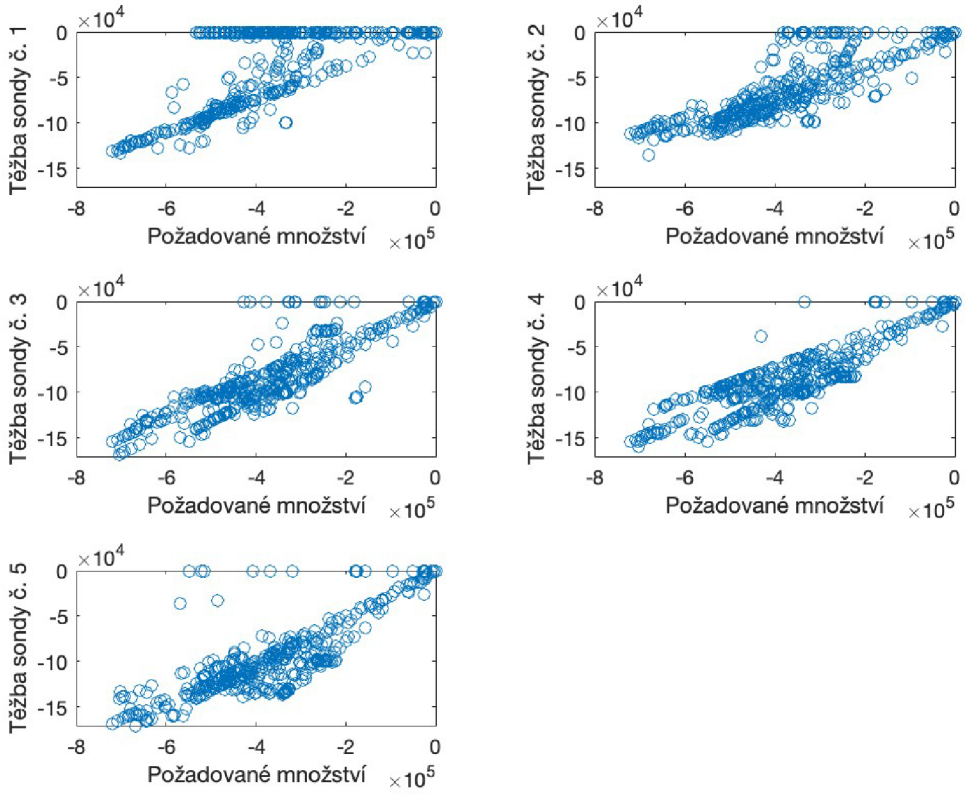
Po rozdělení dat byl z příkazové řádky inicializován objekt 2(b)i s odpovídajícím rozložením dat, který byl následně uložen do souboru „MAT“ (příloha 1(d)iii). Pro použití dat v nadcházejících skriptech bylo nutné pouze načíst tento soubor.

6.4 Analýza dat

Po vyčlenění těžebních dat bylo v práci přistoupeno k jejich analýze. Byly zkoumány závislosti vstupů na výstupech, aby se podložila teoretická tvrzení z kapitoly 2.3. Následně byla provedena analýza četnosti zastoupení jednotlivých vstupů a výstupů. Poté byla definována standardizační metoda, která odstranila měrné rozdíly mezi daty.

6.4.1 Závislost dat

Na obrázku 6.2 je znázorněna závislost těžby z jednotlivých sond na celkovém požadovaném množství (graf obsahuje všechny hodnoty těžebních sezón v jednotkách sm^3). Z grafů lze vyčíst, že existuje míra lineární závislosti mezi celkovým požadovaným množstvím a těžbou konkrétní sondy. Dále je možné zjistit, že sonda č. 1 bývá nejčastěji během těžby zcela uzavřená a celkově se společně se sondou č. 2 podílí na těžbě méně než zbylé sondy, což dokládají i korelační koeficienty pro jednotlivé sondy (viz tabulka 6.3).

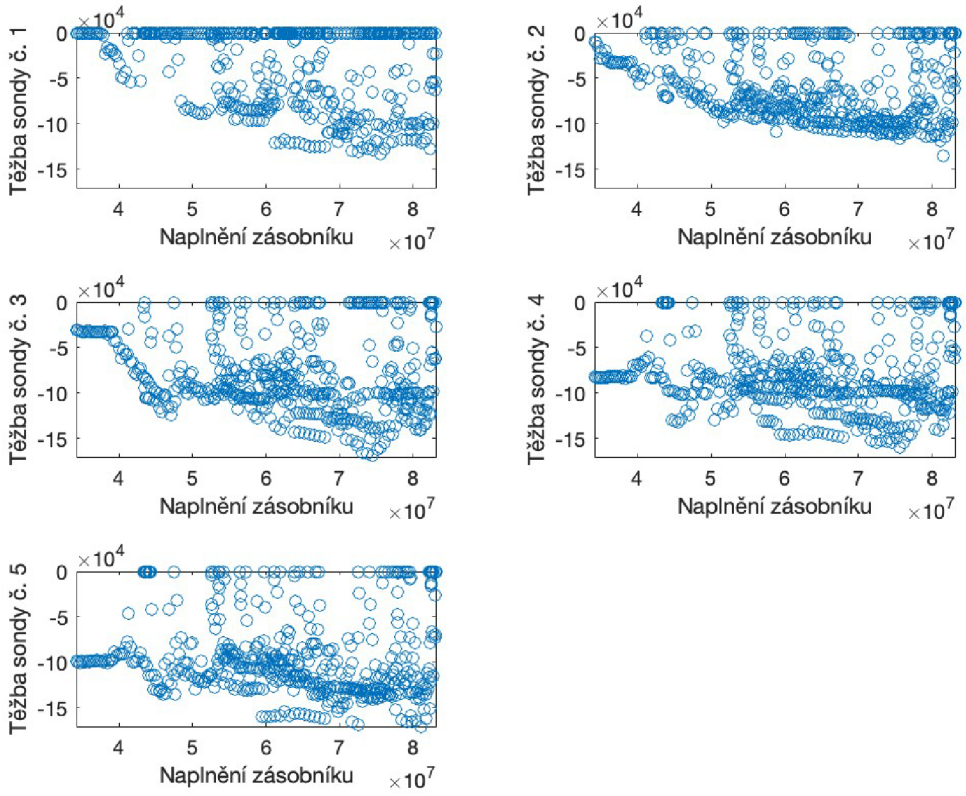


Obrázek 6.2: Závislost požadovaného množství na těžbě z jednotlivých sond

Sonda	1	2	3	4	5
Korelační koeficient	0,71253	0,89512	0,93183	0,95059	0,92824

Tabulka 6.3: Korelační koeficienty požadovaného množství a těžby z jednotlivých sond

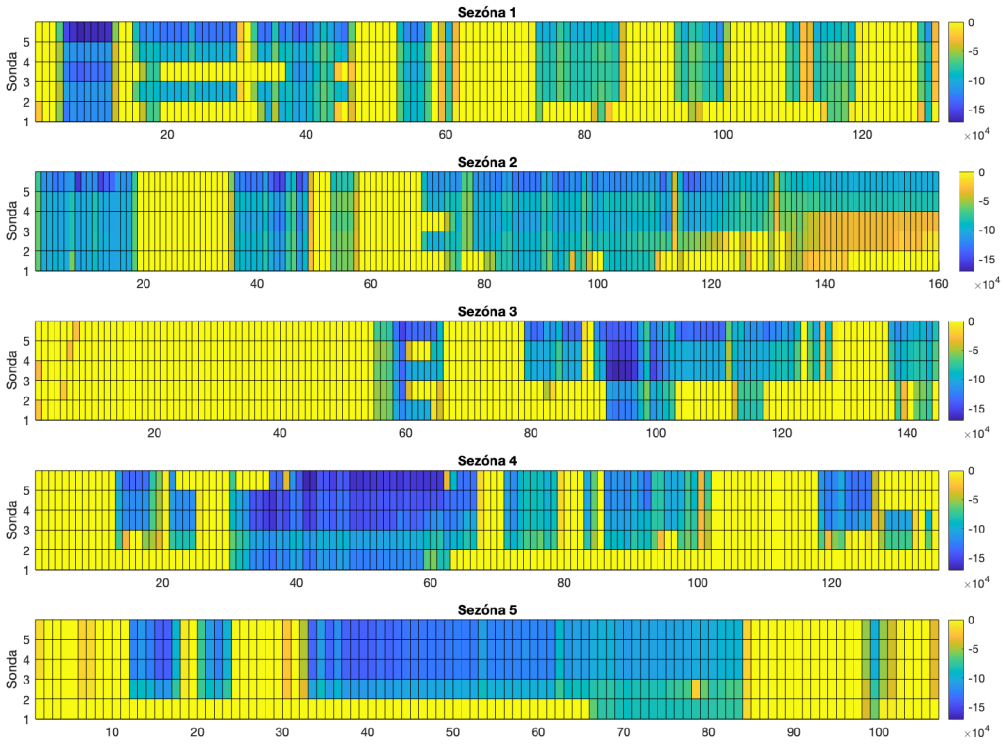
Obrázek 6.3 znázorňuje závislost těžby z jednotlivých sond na stavu zásob v PZP, ze které lze vyčíst, že těžba ze sond č. 1 a 2 postupně klesá při naplněnosti zásobníku menším než cca $5 \times 10^7 sm^3$, následuje sonda č. 3, u které dochází k výraznému snížení těžby po dosažení hranice cca $4,8 \times 10^7 sm^3$. Naopak sondy č. 4 a 5 byly schopny těžit i při nízké naplněnosti zásobníku.



Obrázek 6.3: Závislost stavu zásob v PZP na těžbě z jednotlivých sond

Na obrázku 6.4 jsou znázorněny těžby z jednotlivých sond v průběhu jedné těžební sezóny. Teplost barev značí objem těžby z konkrétní sondy (studenější barva znamená větší objem těžby), na ose x jsou jednotlivé záznamy seřazené dle průběhu času a na ose y jsou vyneseny jednotlivé sondy zásobníku.

V rámci práce nebyly vypozorovány z obrázku žádné indicie, které by objasňovaly závislost rozložení na jednotlivé sondy vzhledem k předešlé těžbě



Obrázek 6.4: Průběh těžby jednotlivých sond v čase

Pro získání výše uvedených informací byl využit skript 2(c)iii. Pro vytvoření grafické podoby těžby z jednotlivých sond (viz obrázek 6.4) byla vytvořena z důvodu dalšího užití v práci samostatná funkce 2(a)i.

6.4.2 Transformace dat

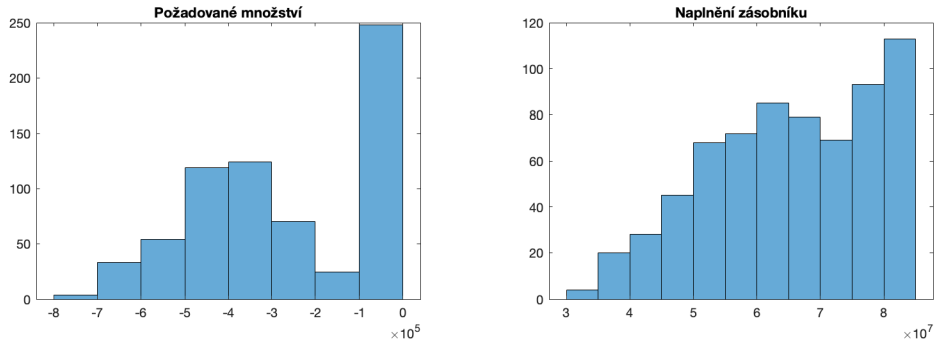
Pro trénování neuronové sítě je vhodné, aby data, ze kterých se síť bude učit, měla pokud možno co nejvíce rovnoměrné rozložení. Rovnoměrným rozložením trénovacích dat se zabrání případnému přeučení sítě na vzorky, které jsou četněji zastoupené v datech.

Při nerovnoměrném rozložení lze použít na data jednu z transformačních metod (například převedení vstupů do logaritmické míry). U použití transfor-

mačních metod se musí brát v úvahu, že transformace data ovlivňují a mohou vést ke ztrátě informace.

Na obrázku 6.5 je znázorněno pomocí histogramů zastoupení jednotlivých vzorků obou vstupů do zamýšlené neuronové sítě. U požadovaného množství plynu ze zásobníku je vidět, že se přibližuje normálnímu rozdělení. Histogram však ukazuje, že v datech je obsaženo velké množství záznamů, ve kterých je těženo méně než sto tisíc sm^3. Následně bylo spočítáno, že z celkového počtu 676 záznamů je 219 s nulovým požadavkem a 29 obsahuje těžbu menší než sto tisíc sm^3. Z povahy řešeného problému a u vědomí poznatků uvedených v kapitolách 2.2 a 2.3 se ale jejich odstranění nepovažovalo za vhodné.

Histogram druhého vstupu (Stavu zásob v PZP) do zamýšlené sítě ukazuje, že je poměrně hodně vzorků do 50 mil sm^3, poté počet zastoupených dat začne výrazně klesat.



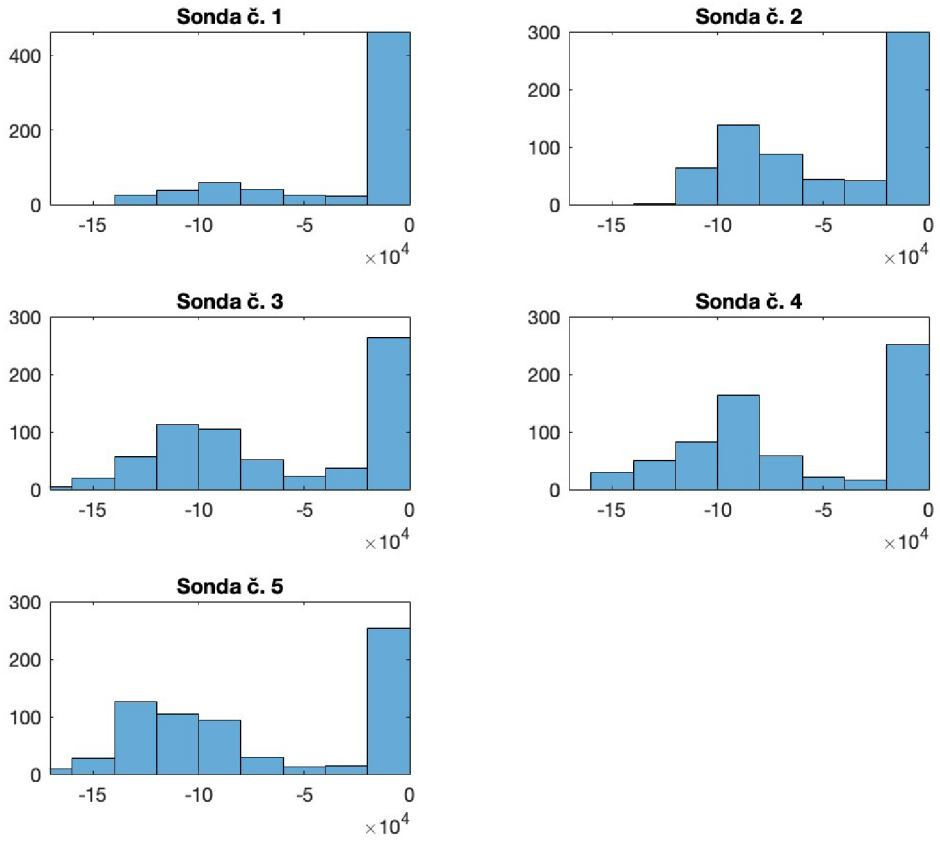
Obrázek 6.5: Histogram dat obou vstupů

U obou vstupů by bylo možné využít logaritmickou, nebo odmocninovou transformační metodu (zachovávají objem těžby i stav zásob a díky výběru pouze dat těžby lze první vstup převést do absolutní hodnoty), ale vzhledem k tvaru histogramu a počtu nulových záznamů u prvního vstupu by ani jedna z těchto metod nebyla účinná.

Na obrázku 6.6 jsou pak znázorněny histogramy pro jednotlivé sondy, na kterých je vidět obdobné rozložení, jaké bylo možné pozorovat u celkového požadovaného množství.

Pro dosažení lepšího rozložení dat bylo vyzkoušeno převedení hodnot na procenta (výstup sítě by představoval rozložení na jednotlivé sondy v procentuálním poměru). Tato myšlenka se ukázala jako nevhodná, protože histogram měl poměrně velké zastoupení do 40 %, pak byla dlouhá prodleva a nakonec bylo několik vzorků v rozmezí 90–100 %.

Poté byly vyzkoušeny logaritmické a odmocninové transformace, které ovšem rovněž nepřinesly výrazné zlepšení v rozložení dat. Z tohoto důvodu nebylo v práci přistoupeno k žádné transformační metodě. Histogramy různě upravených dat si je možné zobrazit pomocí skriptu 2(c)iv.



Obrázek 6.6: Histogramy jednotlivých předloh

6.4.3 Standardizace

Standardizace dat se provádí ze dvou důvodů. Prvním důvodem je nutnost odstranit rozdíl mezi měrnými jednotkami jednotlivých dat. Kdyby nebyly odstraněny, při učení sítě by se nadhodnocovaly váhy u vstupů, které by obsahovaly vyšší číselnou hodnotu. Druhým důvodem je snaha dosáhnout rychlejší konvergence k minimu při procesu učení, když jsou hodnoty procházející sítí v blízkém okolí nuly.

Existuje více metod pro standardizaci dat, mezi nejznámější patří Z-transformace a „min-max“ normalizace. Pro účely této práce byla zvolena metoda „map-min-max“ definovaná vztahem (6.1) s rozsahem $\langle -1, 1 \rangle$ z důvodu snadné implementace při vytváření mělkých sítí v aplikaci Matlab.

$$y = \frac{(ymax - ymin) \cdot (x - xmin)}{(xmax - xmin)} + ymin \tag{6.1}$$

7 MĚLKÁ SÍŤ TYPU NARX

Po přípravě dat bylo možné přistoupit k prvním experimentům trénování neuronové sítě pro daný problém. V počátku řešení této práce bylo zjištěno, že při opakování totožného experimentu (stejná síť i data) se dosahuje odlišných výsledků. Zapříčiňují to dva faktory, které přidávají do algoritmu jednotlivých experimentů náhodnost. První z nich vzniká při automatickém rozdělení dat na trénovací, validační a testovací sadu. Tento faktor nemá vliv na výsledky dosažené v této práci, protože se rozdělení stanovilo předem a je pro všechny experimenty stejné. Druhý faktor výsledky ovlivňuje, nelze jej ale odstranit. Je způsobený náhodným inicializováním vah na začátku trénování sítě a jeho odstranění (například inicializací vah vždy na nulové hodnoty) je při procesu učení neefektivní. Z tohoto důvodu bylo v práci přistoupeno k řešení, v rámci kterého se totožný experiment vždy 5krát zopakuje a výsledek sítě bude průměr RMSE těchto experimentů a jejich směrodatná odchylka σ.

Pro sítě typu NARX je nutné zavést dva níže uvedené pojmy týkající se zpětné vazby, které mají v rámci architektury pro tvorbu mělké sítě v aplikaci Matlab zcela zásadní význam.

Open-loop (otevřená smyčka) je v podstatě síť rozšířená o další vstup ve velikosti zpětné vazby. Pokud je při použití funkce „preparets“ (zmíněné v podkapitole 4.1.1) předána síť s otevřenou smyčkou, funkce vytvoří zpětnou vazbu z předlohových dat a při trénování sítě se použijí data předlohy odpovídající předchozím záznamům jako zpětná vazba.

Close-loop (uzavřená smyčka) znamená skutečnou zpětnou vazbu. Tedy funkce „preparets“ vrátí pouze vstupy odpovídající původním vstupům a při procesu učení se jako zpětná vazba používají skutečné výstupy z předchozího rozhodování sítě.

Sítě typu NARX je tedy možné nejdříve natrénovat pomocí otevřené smyčky, poté ji přetrénovat s uzavřenou smyčkou. Tento proces někdy vede k lepšímu natrénování sítě.

7.1 POTENCIÁLNÍ VÝKON SÍTĚ

První experiment proběhl za účelem stanovení přibližné potenciální hodnoty RMSE pro danou úlohu. Byla tedy natrénovaná síť o 10 neuronech, kdy

zpětnou vazbu tvořily 4 předchozí výstupy (příloha 2(d)i). Aby se dosáhlo potenciální výkonnosti, byla síť natrénována a následně testována s otevřenou smyčkou. Tento přístup neumožňuje získat relevantní výkonnost sítě, protože použití dat předloh ve zpětné vazbě řešený problém výrazně zjednodušuje, ale lze tím dosáhnout hodnoty RMSE, která představuje hranici nejlepšího možného výsledku.

V rámci tohoto experimentu bylo dosaženo průměrné hodnoty RMSE $15\ 555,38\ sm^3$ se směrodatnou odchylkou $\sigma = 1\ 297,67\ sm^3$, kdy trénování sítě skončilo vždy podmínkou maximální validační tolerance (počet validačních výsledků v řadě horších než doposud nejlepší dosažené validační skóre), která byla ve výchozím nastavení rovna šesti.

Výše uvedený experiment byl proveden pouze na datech, která nebyla kromě standardizace upravena. Funkce „preparets“ tedy pro získání zpětné vazby ořezala data o jejich délku, což pro praxi znamená, že síť by byla schopná predikovat rozložení až po n dnech těžby, kde n značí délku zpětné vazby.

Z tohoto důvodu se experiment opakoval s upravenými daty, kde úprava spočívala v přidání prefixu před každou sezónu o délce zpětné vazby (příloha 2(d)ii). Prefix byl zvolený v práci tak, aby simulovat n dnů, kdy nedošlo k těžbě. Obsahoval tedy nulové hodnoty pro vstup požadovaného množství plynu a první hodnotu sezóny pro stav zásob. Předlohy v prefixu byly nulové. Tím se docílilo, že funkce „preparets“ data ořezala pouze o prefix a první hodnota pro trénování byla skutečná hodnota prvního poskytnutého vstupu. Výsledky experimentu jsou znázorněné v tabulce 7.1.

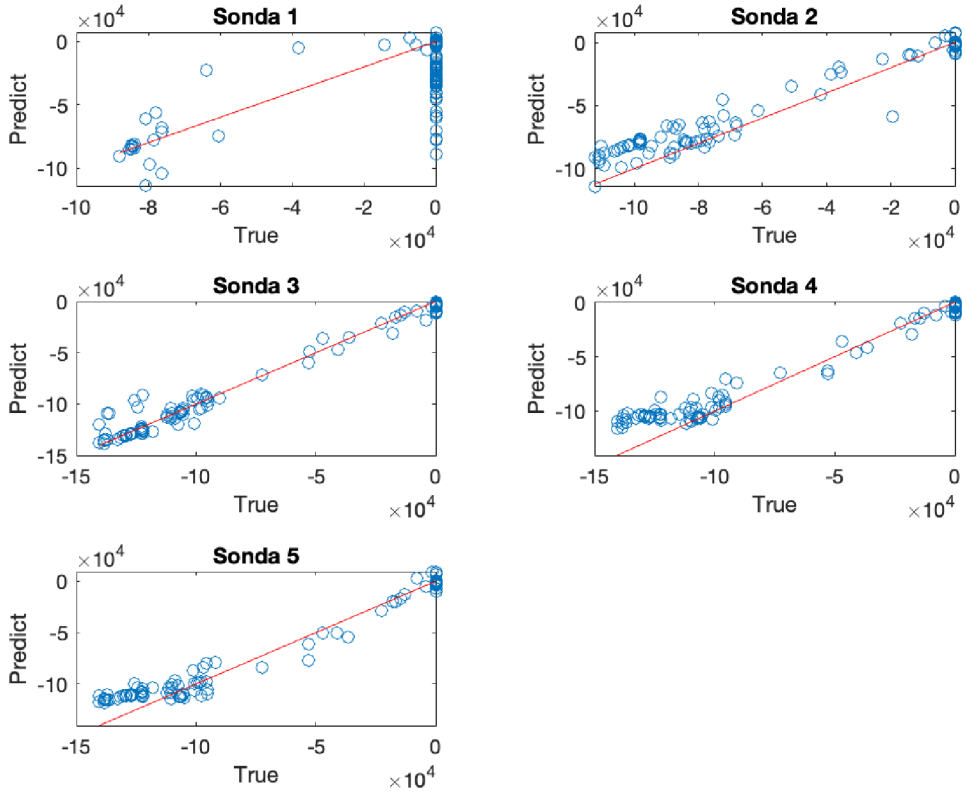
Průměrné RMSE	**Směrodatná odchylka**
$14\ 886,86\ sm^3$	$1\ 265,42\ sm^3$

Tabulka 7.1: Výsledky experimentu z kapitoly 7.1

Na obrázku 7.1 jsou znázorněny grafy regresí pro jednotlivé výstupy z posledního opakování experimentu. Graf regrese znázorňuje, jak jsou predikované hodnoty vzdálené od předlohy. Ideální hodnota (předloha ku předloze) je v grafu znázorněna červenou přímkou, modré body pak značí predikovanou hodnotu (předloha ku predikci). V nejlepším možném případě by tedy všechny modré body ležely na červené přímce.

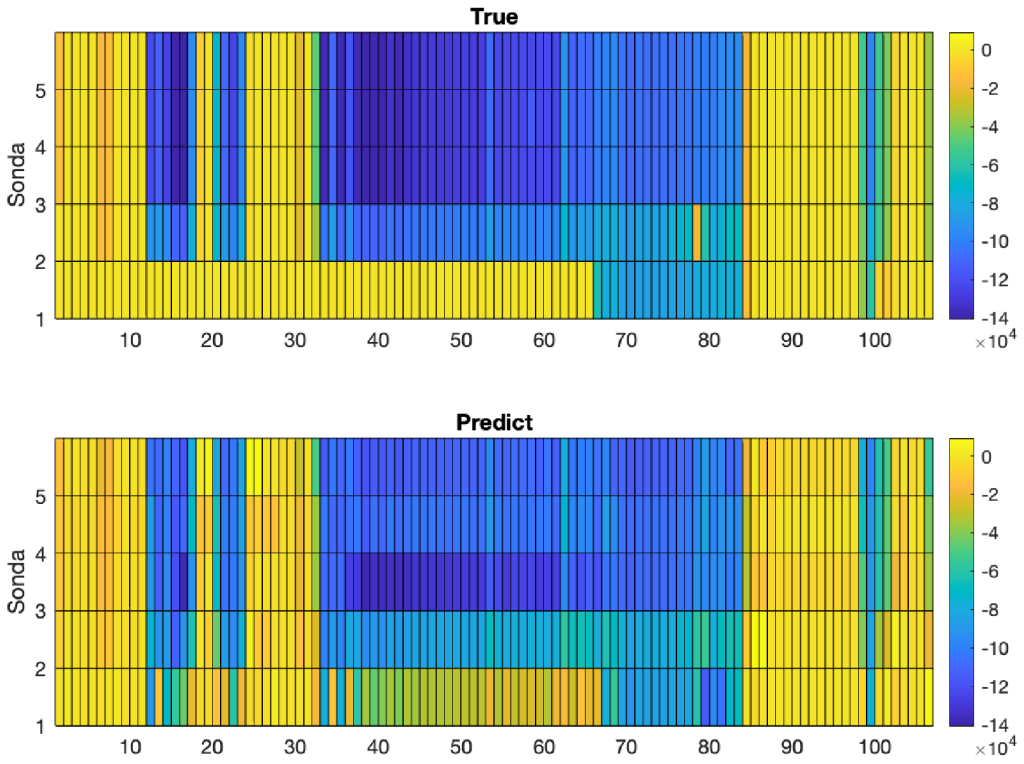
Z jednotlivých grafů je vidět, že se síť naučila poměrně dobře predikovat sondy č. 2–5, ale predikce sondy č. 1 často predikuje poměrně velkou těžbu i pro chtěné nulové hodnoty.

Následně bylo spočítáno, že síť z posledního opakování experimentu predikuje 46krát u jednotlivých sond kladnou hodnotu, která v povaze řešené úlohy představuje vtláčení. Jedná se převážně o hodnoty při nulovém požadavku, kdy žádoucí výstupy jsou také nulové. Zároveň při totožném požadavku lze konstatovat, že výstupy sítě nejsou nikdy nulové, ale pouze se pohybují v nízkých hodnotách přibližujících se nule.



Obrázek 7.1: Regrese potenciální výkonnosti

Na obrázku 7.2 jsou znázorněny předlohy a predikované hodnoty v průběhu času pomocí grafu vysvětleného v podkapitole 6.4.1. Z grafu předlohových dat je možné vyčíst, že při sezóně, která byla vybrána k testování sítě, se sonda č. 1 nepodílela na těžbě až do 66. dne sezóny, ale síť této sondě těžbu přiřazovala. Navíc vždy s největším podílem při větším požadovaném množství plynu, když předchozí dny o délce minimálně stejné, jako je zpětná vazba, byly s nulovým požadavkem. Následující den je podíl u sondy vždy výrazně nižší. To naznačuje, že hodnoty předlohových dat ve zpětné vazbě mají silný vliv, ale přesto síť nepredikuje vypnutí sondy.



Obrázek 7.2: Predikce vs. realita v průběhu času (otevřená smyčka)

7.2 UZAVŘENÁ SMYČKA

Následně bylo provedeno natrénování obdobné sítě z předchozího experimentu, ale již s uzavřenou smyčkou zpětné vazby. Za tímto účelem vznikly dva skripty, které se lišily způsobem trénování.

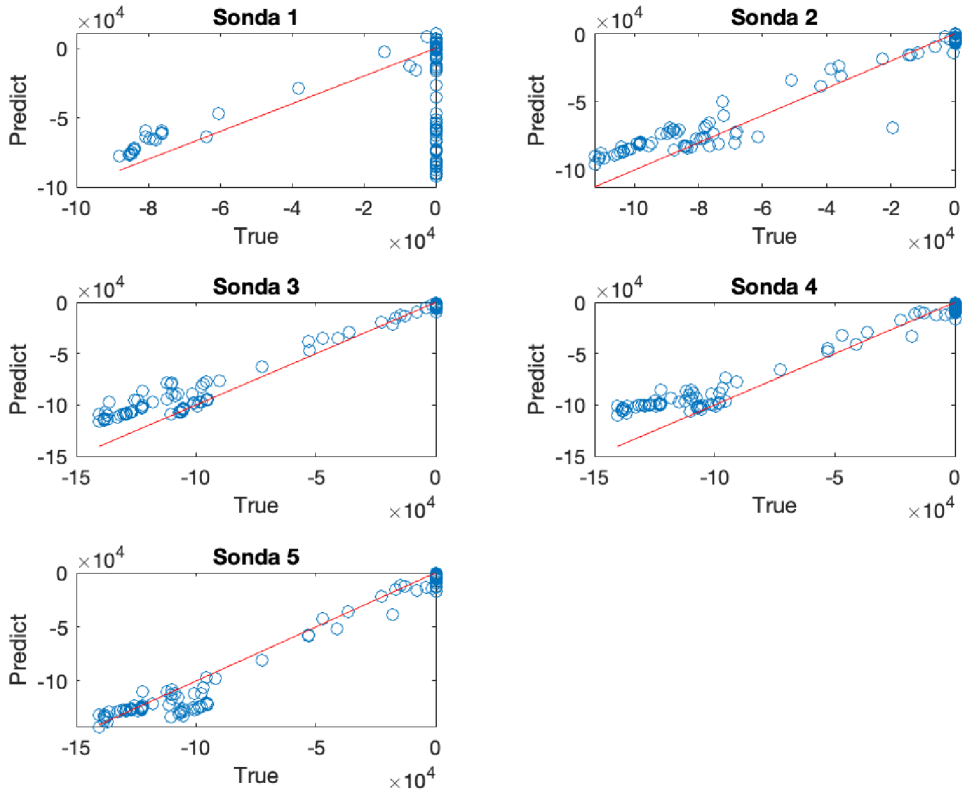
V prvním z nich (příloha 2(d)iii) byla použita metoda předtrénování s otevřenou smyčkou a následného přetrénování sítě s uzavřenou zpětnou vazbou. Ve druhém (příloha 2(d)iv) byla síť natrénována pouze s uzavřenou smyčkou. Výsledky obou metod je možné vidět v tabulce 7.2. Podle průměrného RMSE opakovaných experimentů dosahovala druhá metoda lepších výsledků.

Během pokusů bylo zjištěno, že občas končí trénování sítě s uzavřenou smyčkou po jedné iteraci dosažením maximální hodnoty MU parametru, který se používá k výpočtu gradientu u algoritmu Levenberg-Marquardt zpětné propagace. Tento jev byl vzhledem k nepravidelnostem přiřazován náhodné inicializaci vah při spuštění trénovacího procesu a následného velkého RMSE při prvotním průchodu dat sítí. V rámci práce byly výsledky takto ukončeného experimentu ignorovány a experiment byl opakován.

Metoda	Průměrné RMSE	Směrodatná odchylka
předtrénování	$27\,847,20\ sm^3$	$1\,192,25\ sm^3$
pouze uzavřená smyčka	$25\,537,06\ sm^3$	$657,83\ sm^3$

Tabulka 7.2: Výsledky experimentů z kapitoly 7.2

Na obrázcích 7.3 a 7.4 jsou znázorněny grafy regrese a predikce v průběhu času jednoho z experimentů, kdy trénování proběhlo pouze s uzavřenou smyčkou. Z grafů regrese je pozorovatelné, že síť s uzavřenou zpětnou vazbou poměrně dobře predikuje hodnoty u sond 2–5. Naopak u sondy č. 1 při chtěné nulové hodnotě je predikováno více hodnot s poměrně velkým zastoupením dané sondy na těžbě, než tomu bylo při minulém experimentu s otevřenou smyčkou.

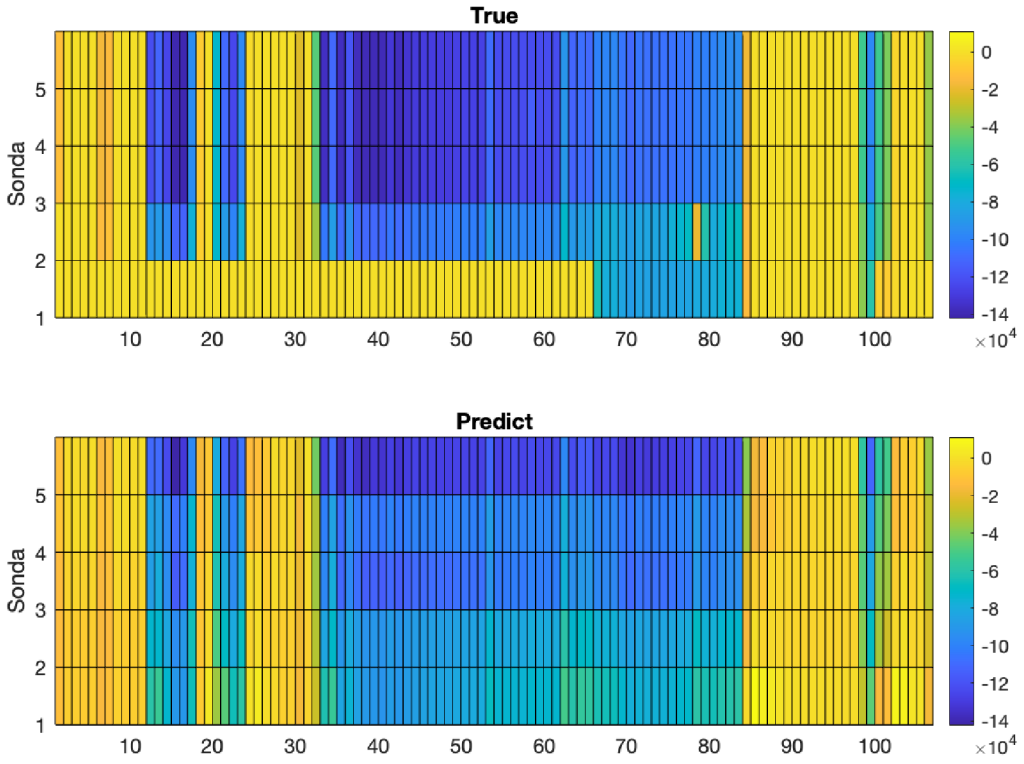


Obrázek 7.3: Regrese uzavřené smyčky

Výše uvedené závěry je možné pozorovat i u druhého ze zmíněných obrázků 7.4, kde síť ignoruje u sondy č. 1 její nečinnost do 66. dne sezóny a predikuje jí konstantně poměrně velký podíl na těžbě.

Z predikovaných hodnot bylo navíc zjištěno, že stále přetrvává problém vzniklý ze znalosti silné závislosti výstupů na požadovaném množství plynu.

Síť se není schopna naučit, že součet výstupů má být roven požadovanému množství a predikované hodnoty mají být menší, nebo rovny nule.



Obrázek 7.4: Predikce vs. realita v průběhu času (uzavřená smyčka)

7.3 OPTIMALIZACE PARAMETRŮ

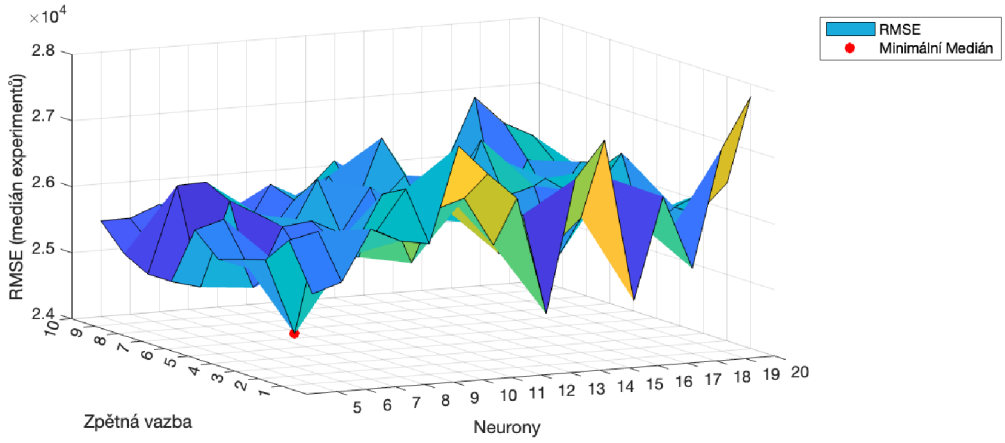
K optimalizaci počtu neuronů ve skryté vrstvě a zpoždění zpětné vazby bylo přistoupeno natrénováním jednotlivých sítí a následným zkoumáním jejich výsledků. Byly natrénovány sítě s 5 až 20 neurony ve skryté vrstvě a zpětnou vazbou o délce 1 až 10 předchozích výstupů sítě.

Jednotlivé experimenty s konkrétní sítí byly provedeny obdobným způsobem jako v předchozí podkapitole 7.2. Vzhledem k tomu, že sítě byly trénovány v programové smyčce, byla pro zajištění ukončení algoritmu nastavena maximální tolerance ignorace trénovacích procesů ukončených parametrem MAX_MU na deset. To znamená, že pokud 10krát během jednoho experimentu neskončilo trénování jinou podmínkou než MAX_MU, pak byl tento experiment ignorován.

Trénování a zaznamenávání jednotlivých výše uvedených sítí bylo provedeno dvěma skripty, které se lišily v módu trénování. V prvním z nich (příloha

2(d)v) bylo použito předtrénování sítě s otevřenou smyčkou a ve druhém (příloha 2(d)vi) byly sítě trénovány pouze s uzavřenou smyčkou. Z důvodu časové náročnosti obou skriptů byly výsledky uloženy do souboru „MAT“ (přílohy 1(d)iv a 1(d)v).

Následně se získaná data zkoumala pomocí skriptu 2(c)v. Na obrázku 7.5 jsou v grafu vyneseny mediány RMSE opakovaných experimentů příslušných sítí (trénovaných pouze se zavřenou smyčkou). Graf ukazuje, že se hodnoty RMSE nejčastěji pohybují nepravidelně mezi 25 až 27 tisíci sm^3. Není ale možné říct, že by ze zvyšování počtu neuronů či délky zpětné vazby vznikala patrná oblast s konvergencí k minimální hodnotě RMSE.



Obrázek 7.5: Medián RMSE opakovaných experimentů

Dále se zkoumaly podrobněji predikované hodnoty sítí s minimálním (6 neuronů a 3 zpětné výstupy) a maximálním RMSE (20 neuronů a 1 zpožděný výstup). Bylo zjištěno, že rozdíl mezi výsledky tvoří převážně hodnoty u nulového požadavku na těžbu, kdy predikce osciluje kolem nízkých hodnot, ale nikdy není zcela nulová. Tento jev potvrdilo i zkoumání výsledků u sítí s okrajovými hodnotami optimalizovaných parametrů (5 neuronů a 1 zpožděný výstup vs. 20 neuronů a 10 zpožděných výstupů).

Na základě výše uvedeného je možné konstatovat, že zvolenou metodou se nepodařilo nalézt optimální síť s konkrétním počtem neuronů ve skryté vrstvě a příslušnou délkou zpětné vazby.

8 VÝSTUPNĚ-VALIDAČNÍ FUNKCE

Dosavadními experimenty bylo zjištěno, že mělká síť typu NARX vytvořená pomocí architektury pro mělké sítě v aplikaci Matlab se není schopna zcela naučit dvě silné závislosti výstupních hodnot na jednom ze vstupů. Protože závislosti jsou známé z daného problému, lze neuronové síti pomoct definicí funkcí řešící tyto závislosti a následně je implementovat v síti.

První známou vlastností je, že vstup x_1 (požadované množství) jasně označuje svým znaménkem těžbu z / vtláčení do PZP. Není tedy přípustné, aby při požadované těžbě byly predikovány u některých sond kladné hodnoty, které v povaze problému znamenají vtláčení. Pro odstranění tohoto jevu je možné po průchodu dat výstupní vrstvou sítě definovat funkci (8.1).

$$f_{Trend}\left(Y_i, x_1\right) = \begin{cases} 0 & \text{když } (x_1 < 0 \wedge Y_i > 0) \vee (x_1 > 0 \wedge Y_i < 0) \vee x_1 = 0 \\ Y_i & \text{když } (x_1 < 0 \wedge Y_i < 0) \vee (x_1 > 0 \wedge Y_i > 0) \end{cases} \tag{8.1}$$

Pro druhou vlastnost, kdy vstup x_1 má být roven součtu jednotlivých výstupních hodnot, může být definována za předpokladu předchozí aplikace vztahu (8.1) funkce (8.2).

$$f_{Sum}\left(Y_i, x_1\right) = \begin{cases} 0 & \text{když } \sum Y = 0 \\ \left(\frac{Y_i}{\sum Y}\right) \cdot x_1 & \text{když } \sum Y \neq 0 \end{cases} \tag{8.2}$$

8.1 VYTVOŘENÍ VRSTEV VALIDACE

Doposud byla v práci použita výhradně architektura popsaná v podkapitole 4.1 pro tvorbu mělké sítě, která však neumožňuje snadnou implementaci vlastních vrstev. Bylo by možné implementovat výše uvedené funkce klasickou Matlab funkcí a po průchodu dat síti je pomocí ní validovat. Tento způsob by ale neumožňoval zapojit validační funkce do procesu učení. Z tohoto důvodu byla namísto architektury pro tvorbu mělkých sítí využita architektura pro tvorbu hlubokých sítí popsané v podkapitole 4.2.

Pro každou z validačních funkcí byla naprogramována z důvodu přehlednosti vlastní mezivrstva, kdy pro správné fungování validace je potřeba použít obě tyto vrstvy za výstupní vrstvou sítě (myšlena vrstva, která transfor-

muje výstupy na chtěný počet, nikoliv výstupní vrstva dle architektury hlubokých sítí, která slouží k trénování sítě), a to v jasně daném pořadí. První musí být zapojena vrstva implementující funkci f_{Trend} (příloha 2(b)iv) a až poté vrstva s funkcí f_{Sum} (příloha 2(b)v).

Z důvodu standardizace dat bylo nutné zajistit, aby se funkcionalita implementovaných funkcí ve vrstvách zachovala i při různých standardizačních metodách. Proto byla data před vykonáním funkce převedena do původních hodnot a po validaci byla opět standardizována. Aby bylo možné data uvnitř vrstev převádět, byla v rámci práce napsána abstraktní třída (příloha 2(b)ii) se dvěma abstraktními funkcemi: „standardize“ a „destandardize“. Pro standardizaci dat pak musí být implementována třída, která dědí z této abstraktní třídy (standardizaci metodou „map-min-max“ vykonává objekt 2(b)iii). Tyto standardizační objekty se následně předávají konstruktoru obou vrstev.

8.1.1 Zpětný průchod

Neuronová síť je v podstatě složení několika funkcí (jednotlivých vrstev), kdy se dopředným průchodem aplikují hodnoty od nejvíc vnořené funkce (první vrstvy) až po nejsvrchnější funkci (poslední vrstva). Algoritmus zpětné propagace vypočítá gradient, který se poté propaguje derivací složené funkce. Jelikož se jedná vždy o složenou funkci, je využito řetízkové pravidlo.

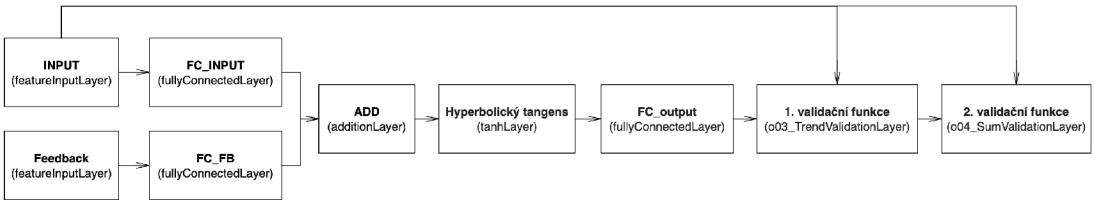
Metoda pro zpětnou propagaci obou vlastních vrstev tedy implementuje níže uvedené parciální derivace příslušných funkcí, které jsou násobeny hodnotami gradientu ($dLdZ$) vypočítaného níže položenou vrstvou.

$$
\begin{aligned}
\frac{\partial f_{Trend}}{\partial Y_i} &= \begin{cases} 0 & \text{když } (x_1 < 0 \wedge Y_i > 0) \vee (x_1 > 0 \wedge Y_i < 0) \vee x_1 = 0 \\ dLdZ & \text{když } (x_1 < 0 \wedge Y_i < 0) \vee (x_1 > 0 \wedge Y_i > 0) \end{cases} \\
\frac{\partial f_{Trend}}{\partial x_1} &= 0 \\
\frac{\partial f_{Sum}}{\partial Y_i} &= \begin{cases} 0 & \text{když } \sum Y = 0 \\ \left(\frac{x_1}{\sum Y}\right) \cdot dLdZ & \text{když } \sum Y \neq 0 \end{cases} \\
\frac{\partial f_{Sum}}{\partial x_1} &= \begin{cases} 0 & \text{když } \sum Y = 0 \\ \left(\frac{Y_i}{\sum Y}\right) \cdot dLdZ & \text{když } \sum Y \neq 0 \end{cases}
\end{aligned}
\tag{8.3}
$$

Funkce zpětné propagace obdobně jako při dopředném průchodu převede data do původní podoby a ta se po výpočtu gradientu standardizují. Vzhledem k tomu, že vstup do vrstev x_1 představuje reálný vstup do sítě a nemá tedy vliv na další šíření gradientu sítí, byly do zpětné propagace předány pouze nulové hodnoty a reálný výpočet derivace byl u vstupu x_1 ignorován.

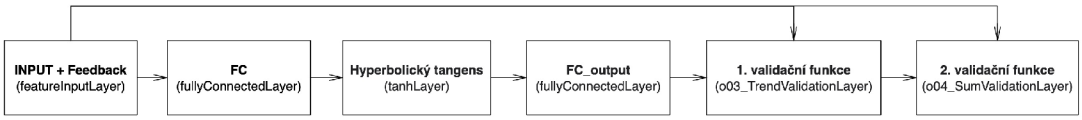
8.2 IMPLEMENTACE SÍTÍ

V rámci práce byly nejdříve vytvořeny dvě architektury sítě, ve kterých byly implementovány vlastní validační vrstvy. Tyto architektury se liší ve zpracování vstupu, kdy první architektura, znázorněná na obrázku 8.1, obsahuje dvě vstupní vrstvy a k nim příslušné plně propojené vrstvy.



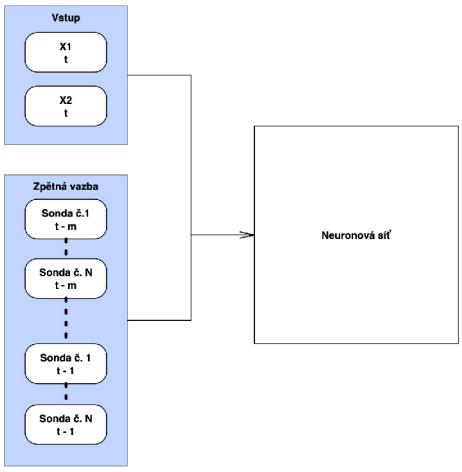
Obrázek 8.1: První architektura sítě s validačními vrstvami

Následně bylo výše uvedené schéma zjednodušeno tím, že byla zpětná vazba přímo přidána ke vstupům do sítě a byla použitá pouze jedna plně propojená vrstva (viz obrázek 8.2).



Obrázek 8.2: Druhá architektura sítě s validačními vrstvami

Vytvoření zpětné vazby, kdy vstupní vrstva sítě je tvořená vrstvou pro příznaky „featureInputLayer“, bylo provedeno dle schématu na obrázku 8.3 přidáním dalších vstupních hodnot do sítě, a to následujícím způsobem. Pro vstupy do sítě v čase t byly přidány další vstupní hodnoty o velikosti $n \cdot m$ (kde n značí počet výstupů sítě a m je délka zpětné vazby), které obsahují hodnoty předchozího rozhodování sítě. Řazení vektoru obsahujícího jednotlivé zpětné vazby vzhledem k času zpoždění je sestupné (nejméně zpožděná zpětná vazba je řazena do posledních řádků). Jednotlivé výstupy (sondy) jsou vzhledem k jejich pořadí řazeny vzestupně (sonda č. 1 je na prvním řádku příslušné zpětné vazby).



Obrázek 8.3: Zpracování zpětné vazby příznakové vstupní vrstvy

8.2.1 Otevřená smyčka

Výše uvedené sítě byly vytvořeny, natrénovány a testovány s otevřenou smyčkou pomocí skriptů 2(d)vii a 2(d)viii. Data byla upravena prefixem a následně standardizována stejným způsobem jako při předchozích experimentech s architekturou pro mělké sítě (kapitola 7).

U každé ze sítí byla využita velikost deseti neuronů ve skryté plně propojené vrstvě. To znamená, že u první uvedené sítě bylo deset neuronů ve skryté vrstvě pro vstupní data a stejný počet byl i ve skryté vrstvě pro zpětnou vazbu. Druhá architektura obsahovala pouze jednu plně propojenou vrstvu o deseti neuronech.

Vzhledem k otevřené smyčce bylo možné využít trénovací funkce „trainNetwork“, jíž byly předány konfigurace uvedené v tabulce 8.1.

Optimalizační metoda	Adam
Inicializační míra trénování	0,01
Promíchání dat	každou epochu
Validační tolerance	6
Výstupní síť	nejlepší validační výsledek

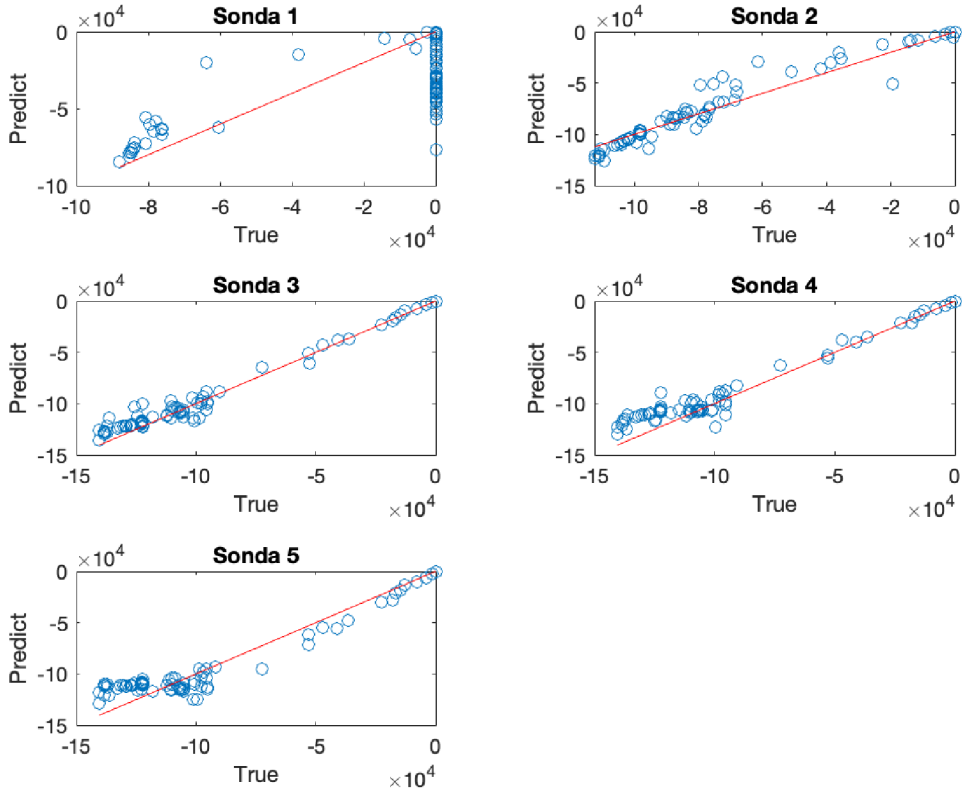
Tabulka 8.1: Nastavení trénování sítí z kapitoly 8.2.1

Obě sítě dosáhly lepších výsledků než síť z kapitoly 7.1. Architektura s odděleným vstupem pro zpětnou vazbu dosáhla lepších výsledků než druhá z architektur. Možnou příčinou rozdílu je dvojnásobný počet neuronů ve skryté vrstvě ($2 \cdot 10$). Hodnoty výkonu obou sítí jsou uvedeny v tabulce 8.2. Níže

pak lze na obrázku 8.4 vidět grafy regresí posledního experimentu druhé architektury.

Síť	Průměrné RMSE	Směrodatná odchylka
Oddělená zpětná vazba	$11\ 989,61\ sm^3$	$1\ 044,47\ sm^3$
Jeden společný vstup	$12\ 925,03\ sm^3$	$1\ 188,05\ sm^3$

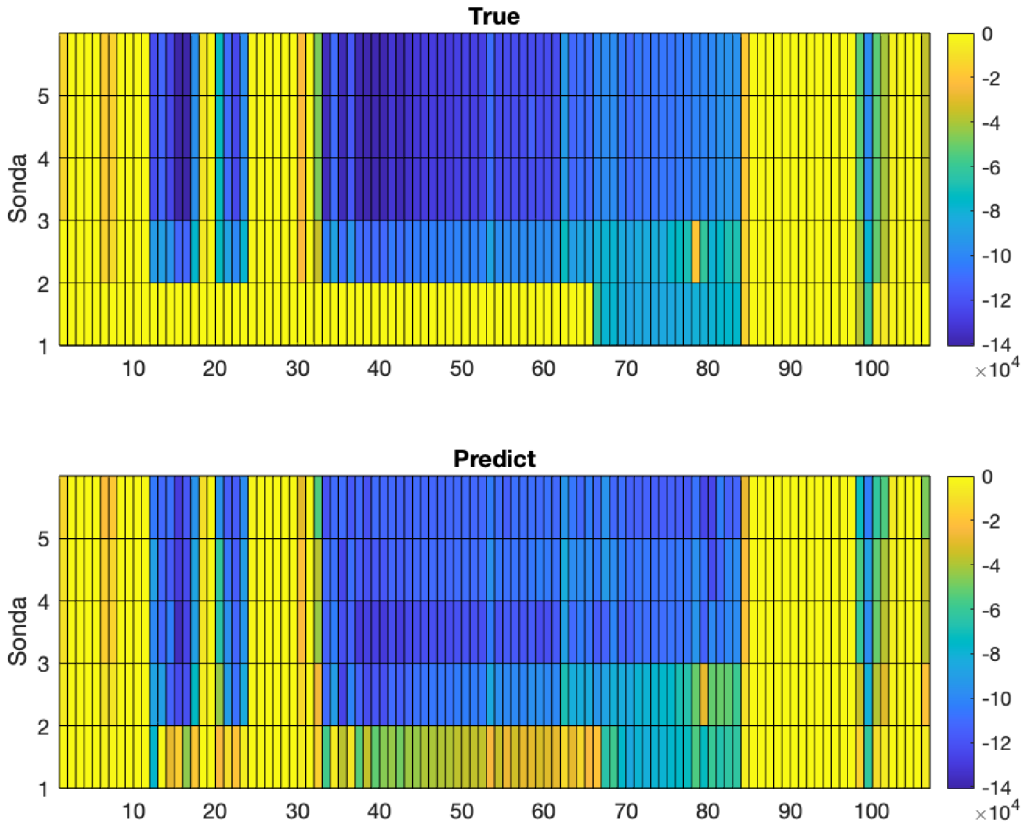
Tabulka 8.2: Výsledky experimentů sítí z kapitoly 8.2.1



Obrázek 8.4: Regrese sítě z kapitoly 8.2.1

Graf regresí a výsledky v průběhu času (obrázek 8.5) ukazují, že validační funkce fungují. U vzorků, kde je požadované množství nulové, jsou rovněž nulové i všechny výstupní hodnoty a žádná sonda nepredikuje vtláčení. Dále je na obrázku vidět, že síť není schopna opět predikovat nepodílení se sondy č. 1 na těžbě do 66. dne sezóny a výsledky jsou snižovány předlohami ve zpětné vazbě.

Z výše uvedeného lze konstatovat, že zavedení validačních funkcí poměrně výrazně zlepšilo výsledky, ale síť stále ignoruje vynechání sondy č. 1 v první polovině sezóny.



Obrázek 8.5: Predikce vs. realita v průběhu času z kapitoly 8.2.1

8.2.2 Uzavřená smyčka

Sítě s uzavřenou smyčkou vytvořené architekturou pro tvorbu hlubokých sítí nelze trénovat pomocí funkce „trainNetwork“, protože ta nedisponuje zpracováváním výstupů během trénování. Z tohoto důvodu pro tyto sítě byla implementována vlastní tréninková smyčka.

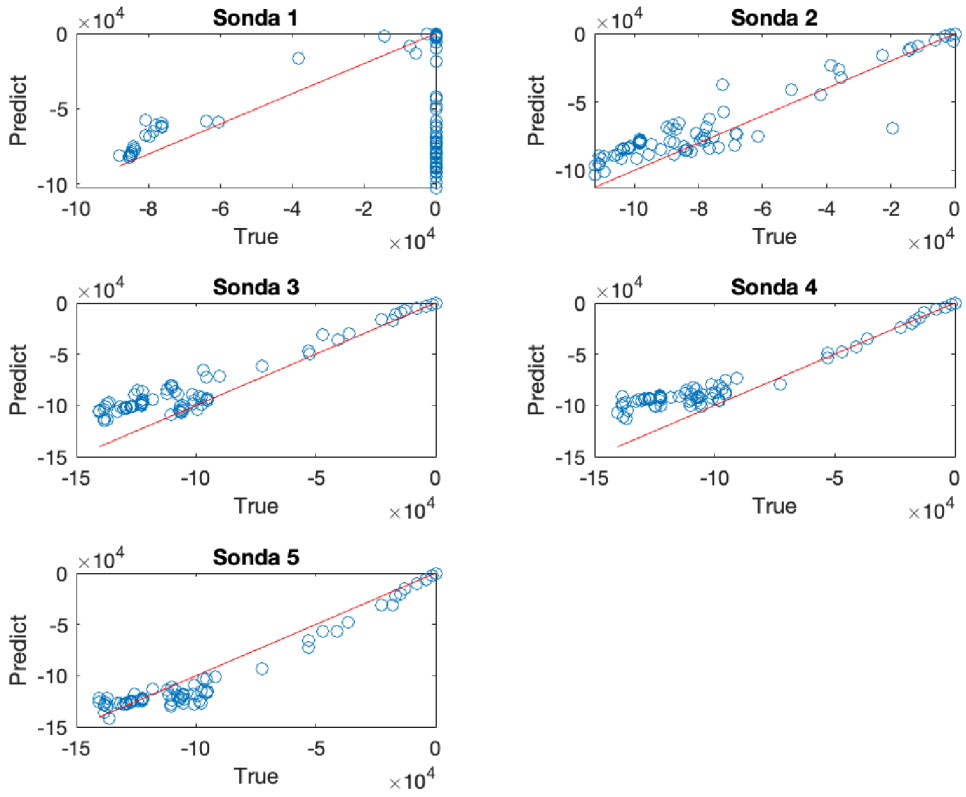
Sítě byly vytvořeny, trénovány a testovány pomocí skriptů 2(d)ix a 2(d)x. V tréninkové smyčce byla použita jako minidávka celá jedna sezóna a data se během tréninku nepromíchávala. Změna hodnot ve zpětné vazbě byla provedena vždy na konci epochy a síť byla trénována bez předtrénování s otevřenou smyčkou. Pro výpočet a propagaci gradientu byla využita obdobně jako při trénování pomocí funkce optimalizační metoda Adam.

Z výsledků uvedených v tabulce 8.3 vyplývá, že obě architektury se pohybují zhruba ve stejné výkonnosti a jsou srovnatelné s výsledky dosaženými v kapitole 7.2. Je ale nutné připomenout, že RMSE zohledňuje pouze vzdálenost predikce od předlohy a nezáleží na tom, jestli jsou predikované hodnoty větší, nebo menší než předlohy. Plný efekt validačních funkcí tedy RMSE nezohledňuje.

Síť	Průměrné RMSE	Směrodatná odchylka
Oddělená zpětná vazba	$26\ 205,08\ sm^3$	$1\ 282,82\ sm^3$
Jeden společný vstup	$25\ 788,90\ sm^3$	$822,64\ sm^3$

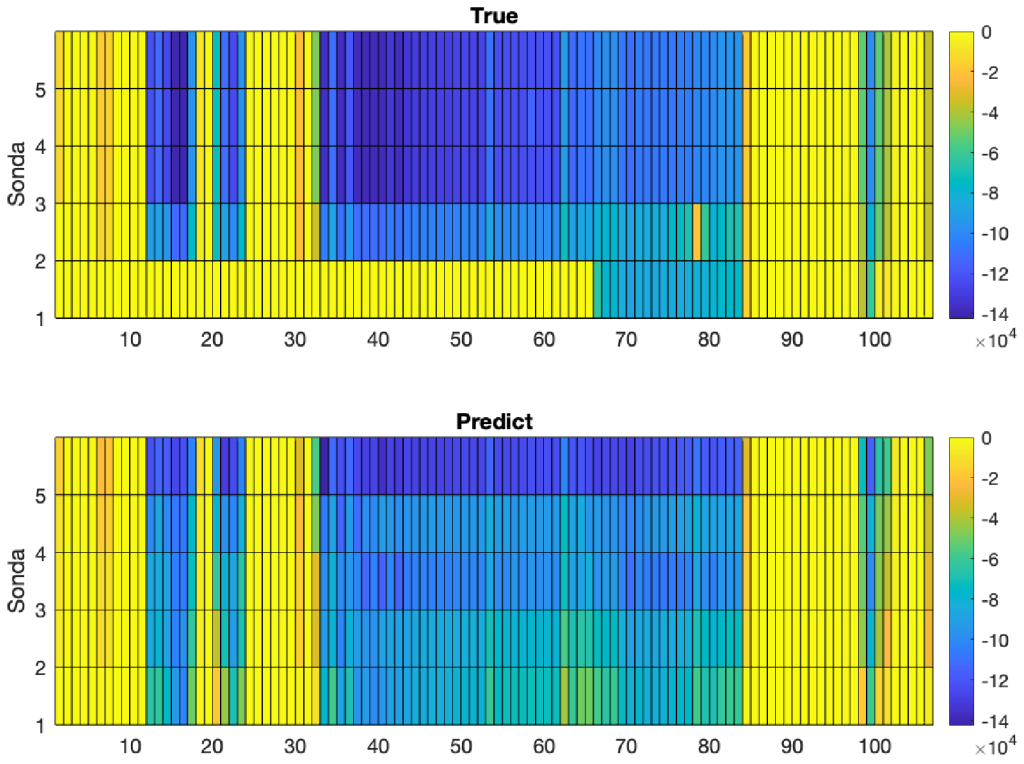
Tabulka 8.3: Výsledky experimentů sítí z kapitoly 8.2.2

Níže jsou opět uvedeny grafy regresí (obrázek 8.6) a graf predikce vůči realitě v průběhu času (obrázek 8.7), na němž je možné pozorovat podobné chování jako v experimentu kapitoly 7.2 s tím rozdílem, že je zde jasně viditelný vliv validačních funkcí.



Obrázek 8.6: Regrese sítě z kapitoly 8.2.2

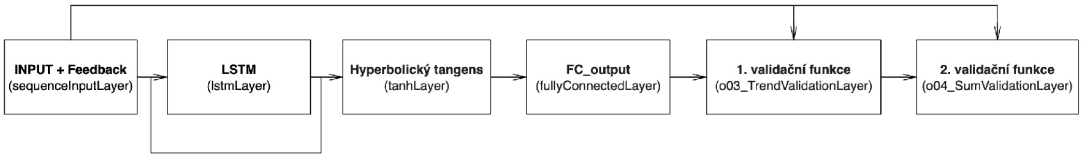
Na základě výše uvedených informací lze konstatovat, že validační funkce pomohly síti zlepšit predikci tím, že již žádné sondy nepredikují vtláčení a součet výstupů je vždy roven požadovanému množství. Potenciální výkonnost u sítí s validačními funkcemi byla lepší než u předchozího obdobného experimentu, ale výkony dle RMSE s uzavřenou smyčkou byly srovnatelné.



Obrázek 8.7: Predikce vs. realita v průběhu času z kapitoly 8.2.2

9 REKURENTNÍ VRSTVA

Z důvodu, že řešený problém odpovídá úlohám časových řad, pro které existují speciální rekurentní vrstvy, byla v práci vytvořena ještě jedna architektura sítě (znázorněná na obrázku 9.1). Skrytá plně propojená vrstva z druhé architektury předchozího experimentu byla nahrazena rekurentní vrstvou pro učení dlouhodobé závislosti mezi časovými kroky (LSTM vrstva). Protože rekurentní vrstvy pracují se sekvenčními daty, byla změněna i vstupní vrstva na sekvenční.

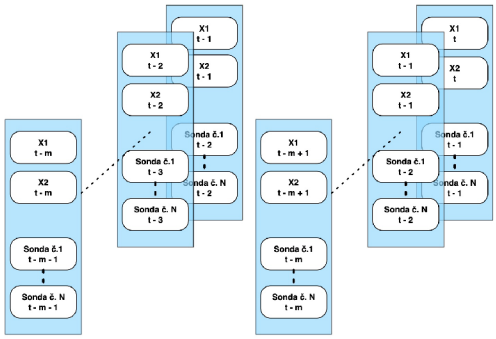


Obrázek 9.1: Architektura sítě s LSTM vrstvou

Jako první byla vyzkoušena myšlenka, že jedna sekvence bude obsahovat celou jednu sezónu. To se ale po prvních pokusech (příloha 2(d)xi) ukázalo jako nevhodné, protože síť se pak učila pouze ze třech dlouhých sekvencí.

Následně bylo přistoupeno k tvorbě sekvence o zvolené délce zpětné vazby (LSTM vrstva se bude učit pouze závislosti z několika předchozích časových stop). Skládání dat do sekvence je znázorněno na obrázku 9.2, kdy vstupní data pro okamžik t byla tvořena aktuálními vstupy a výstupy z předchozího kroku $t-1$. Jedna sekvence pak byla tvořena takovými záznamy m, kde m je délka zpětné vazby. Vzorky sekvencí byly tvořeny posouváním časové řady vždy o jeden krok t. Pro níže uvedené experimenty byla zvolena sekvence o délce čtyř zpětných vzorků.

U LSTM vrstvy je možné volit mezi dvěma formáty výstupních dat. Vrstva dále posílá data buď ve formátu celé příchozí sekvence, nebo pouze výstupu pro poslední časový úsek. Výše uvedené vytvoření sekvence počítá právě s druhým zmiňovaným výstupem vrstvy a toto chování je nutné zadat při její inicializaci.



Obrázek 9.2: Zpracování zpětné vazby sekvenční vstupní vrstvy

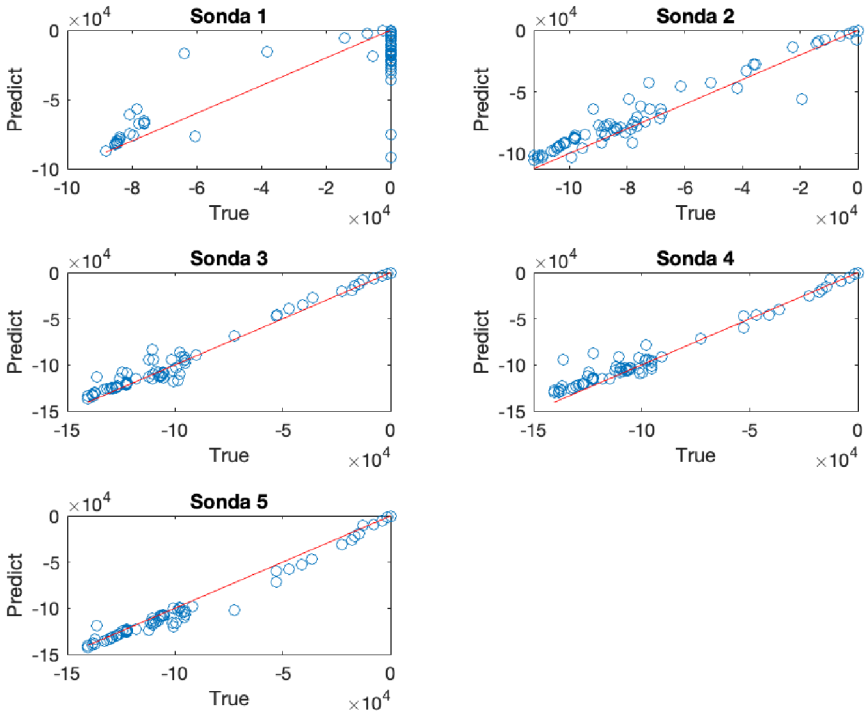
9.1 OTEVŘENÁ SMYČKA

V práci byla nejprve jako obvykle vyzkoušena síť s otevřenou smyčkou zpětné vazby, která byla nejdříve natrénována pomocí funkce „trainNetwork“ se stejným nastavením jako u experimentu z kapitoly 8.2.1 (skript 2(d)xii). Trénování tímto způsobem z doposud nezjištěných důvodů vykazovalo nezvykle vysoké hodnoty dosažitelného RMSE trénovacích a validačních dat během procesu učení. Ten byl pak ukončen po několika epochách dosažením validační tolerance. Z tohoto důvodu byla síť natrénována i pomocí vlastní tréninkové smyčky (příloha 2(d)xiii), kde výsledky trénovacích dat během trénování stabilně konvergovaly k minimální hodnotě RMSE. Výkony sítí trénovaných pomocí obou přístupů jsou uvedeny v tabulce 9.1.

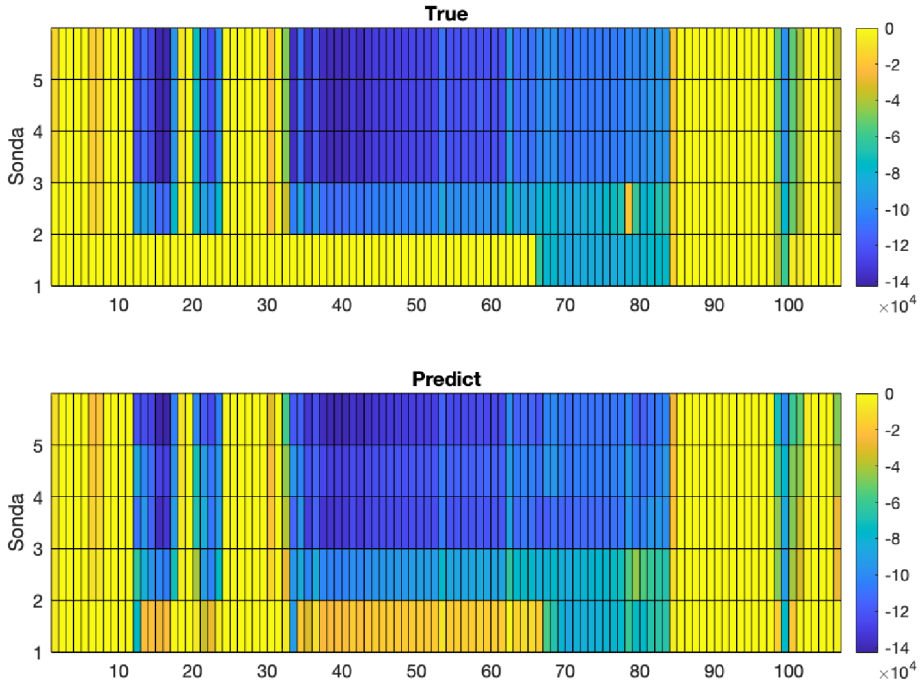
Trénování	Průměrné RMSE	Směrodatná odchylka
trainNetwork	$15\,338,75\ sm^3$	$3\,567,62 sm^3$
vlastní smyčka	$11\,373,23\ sm^3$	$931,43\ sm^3$

Tabulka 9.1: Výsledky experimentů sítí z kapitoly 9.1

Z výše uvedených výsledků je zřejmé, že při trénování s vlastní smyčkou bylo dosaženo nepatrně lepších výsledků než při experimentu z kapitoly 8.2.1. Níže jsou uvedeny grafy regresí a predikce v průběhu času sítě natrénované vlastní smyčkou, kde lze vidět rovněž zlepšení, ale současně také neustálou tendenci přiřazovat sondě č. 1 těžbu v počátku sezóny, kdy je uměle srážena hodnotou předloh ve zpětné vazbě.



Obrázek 9.3: Regrese sítě z kapitoly 9.1



Obrázek 9.4: Predikce vs. realita v průběhu času z kapitoly 9.1

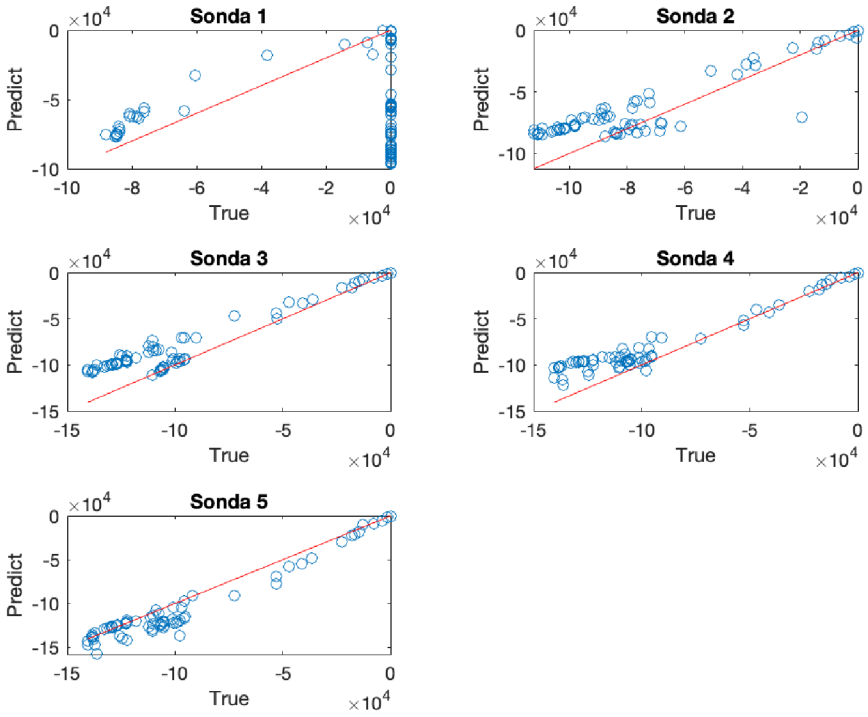
9.2 UZAVŘENÁ SMYČKA

Algoritmus trénování s uzavřenou smyčkou byl proveden obdobným způsobem jako u experimentu v kapitole 8.2.2 a experimenty této sítě byly provedeny pomocí skriptu 2(d)xiv. Z níže uvedených výsledků v tabulce 9.2 je evidentní, že síť dosahuje podobné výkonnosti jako u experimentu v kapitole 8.2.2.

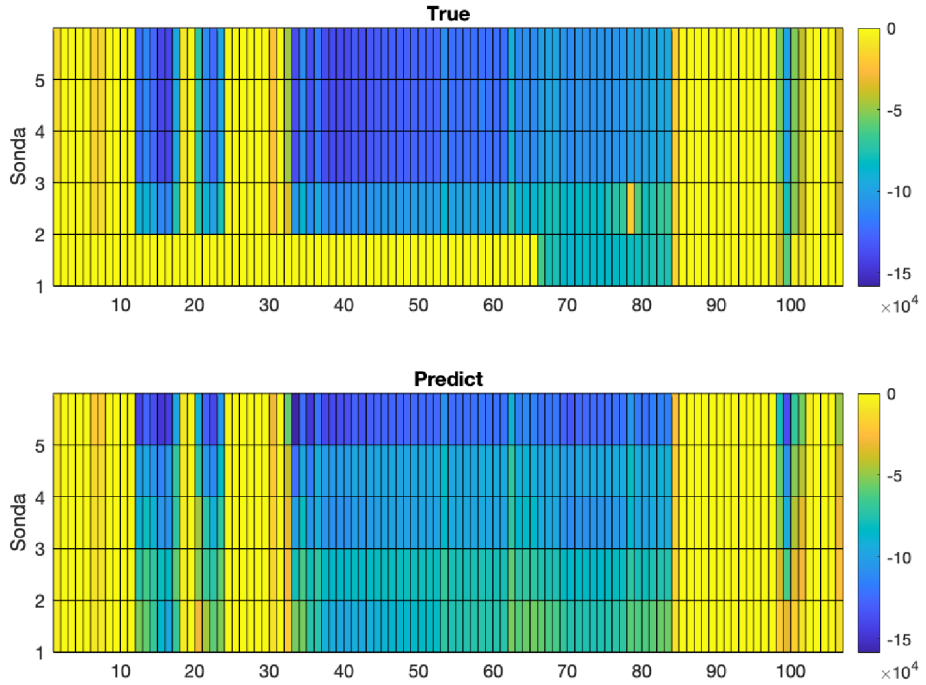
Průměrné RMSE	**Směrodatná odchylka**
$26\,286,59\ sm^3$	$917,26\ sm^3$

Tabulka 9.2: Výsledky experimentů sítě z kapitoly 9.2

Na grafu regrese vidíme podhodnocování sond č. 2–4, které síť kompenzuje sondou č. 1. Tento jev byl pozorovatelný u všech provedených experimentů sítí s uzavřenou smyčkou zpětné vazby. Nyní lze tedy říct, že ani jedna ze sítí si v rámci této práce s tímto problémem nedokázala poradit a je na místě uvažovat o možné technické odstávce sondy č. 1 na začátku sezóny, která byla použita pro testování. Pokud by opravdu sonda č. 1 byla odstavena z technických důvodů, pak by se tento jev mohl odstranit přidáním dalších vstupů do sítě, které by obsahovaly jednoduchou informaci o tom, zda se konkrétní sonda může podílet na těžbě, či nikoliv. Síť by se pak mohla naučit i rozdělení na sondy v případě, kdy se některé nemohou podílet na těžbě.



Obrázek 9.5: Regrese sítě z kapitoly 9.2



Obrázek 9.6: Predikce vs. realita v průběhu času z kapitoly 9.2

10 POROVNÁNÍ SE SIMULÁTOREM

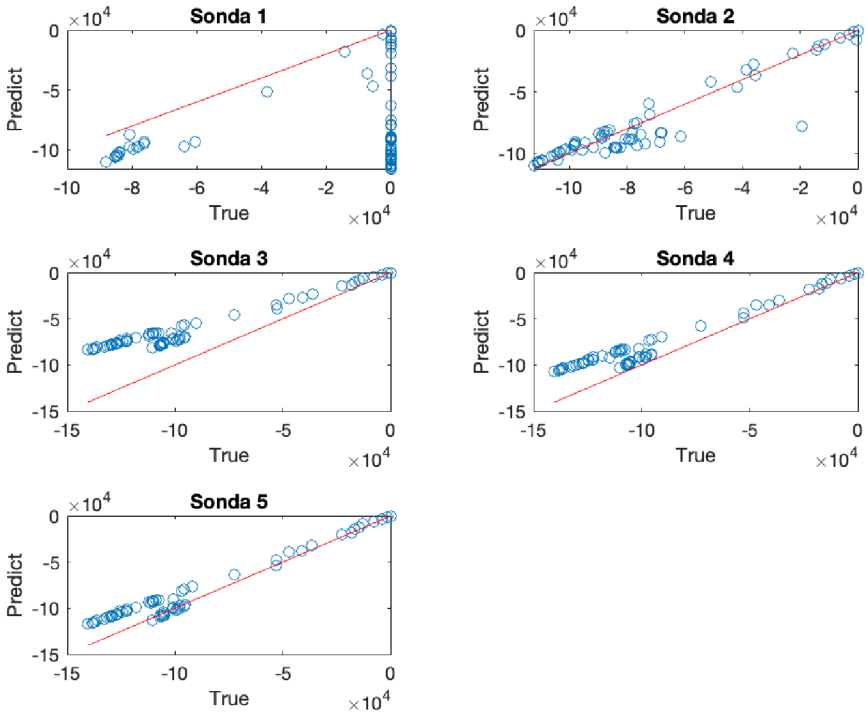
V rámci práce byly poskytnuty výstupy ze simulátoru pro všechny použité záznamy (příloha 1c). Z poskytnutých dat byla vybrána pouze ta data, která odpovídala testovací sezóně. Následně bylo spočítáno RMSE a zobrazeny grafy regresí jednotlivých sond a graf predikce vůči realitě v průběhu času obdobně, jak tomu bylo při testování sítí. Uvedené činnosti byly provedeny pomocí skriptu 2(c)vi.

Hodnota RMSE pro výstupy ze simulátoru činí $34\ 726,25\ sm^3$, což je přibližně o $8\ 500\ sm^3$ horší výsledek, než dosahovala síť z kapitoly 9.2. Dále je možné z obrázků 10.1 a 10.2 pozorovat, že simulace rovněž ignoruje vypnutí sondy č. 1 v první polovině testovací sezóny. Na rozdíl od poslední sítě ale kompenzuje činnost sondy č. 1 menším vytížením sond č. 3–5, přičemž poslední síť kompenzovala tuto anomálii sondami č. 2–4.

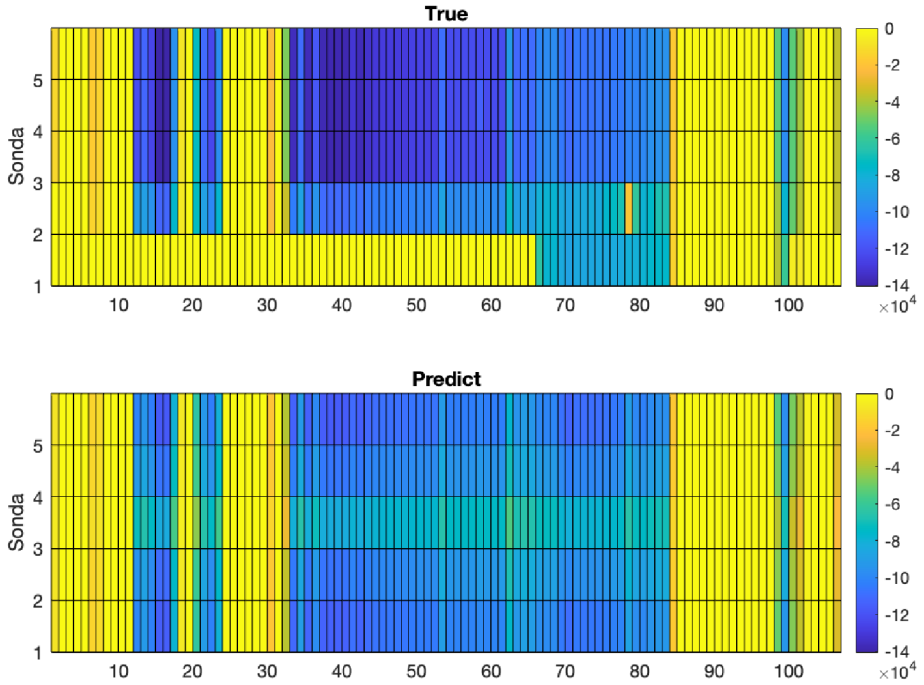
Pro možnost srovnání výpočetní náročnosti sítí se simulátorem bylo přistoupeno k měření času trvání průběhu kódu. U jednotlivých sítí byla změřena doba trénování a testování (hodnoty jsou uvedeny v úvodních komentářích příslušných skriptů). V naměřených hodnotách je možné pozorovat poměrně velké časové skoky, jejichž odůvodnění lze vyčíst z tabulky 10.1.

Architektura	**Metoda trénování**	**Metoda testování**	**Doba trénování [s]**	**Doba testování [s]**
Mělké sítě	train()	Hromadná simulace	0,88–2,86	0,01–0,05
Hluboké sítě	trainNetwork()	Hromadná simulace	13–20	0,11–0,26
Hluboké sítě	Vlastní smyčka	Jednotlivé záznamy	62–117	0,64–0,68

Tabulka 10.1: Časová náročnost trénování a testování sítí



Obrázek 10.1: Regrese výstupů ze simulátoru



Obrázek 10.2: Predikce vs. realita v průběhu času výstupů ze simulátoru

11 ZÁVĚR

V rámci práce byly natrénovány na poskytnutých provozních datech jednoho PZP čtyři architektury sítě typu NARX, přičemž u každé z architektur byly provedeny experimenty s otevřenou i uzavřenou smyčkou zpětné vazby.

První z výše uvedených architektur byla tvořena jednou plně propojenou skrytou vrstvou a výstupní vrstvou, kdy síť byla vytvořena nástrojem pro mělké sítě v aplikaci Matlab. Výsledky experimentů této architektury ukázaly, že síť se není schopna zcela dobře naučit dvě silné závislosti plynoucí z povahy řešeného problému. Výstupy ze sítě se nikdy nerovnaly požadovanému množství plynu, a dokonce se u několika málo výstupů predikovalo vtláčení. Toto zjištění vedlo k definování „výstupně-validačních“ funkcí, které tyto závislosti řeší. Aby bylo možné tyto funkce vytvořit jako vrstvy sítě a zapojit je přímo do procesu učení, došlo ke změně nástroje a následné sítě byly vytvořeny pomocí architektury pro hluboké sítě aplikace Matlab. To s sebou přineslo nutnost vlastního řešení zpětné vazby, a tak vznikly další dvě architektury sítí. První z nich měla oddělený vstup pro zpětnou vazbu se svou vlastní plně propojenou skrytou vrstvou. Druhá vznikla za účelem přiblížit se řešení zpětné vazby z architektury pro mělké sítě, tedy mít pouze zpětnou vazbu jako další maticové vstupy do jedné plně propojené vrstvy. U obou těchto sítí byly připojeny za výstupní vrstvu vlastní „výstupně-validační“ vrstvy. Výsledky experimentů ukázaly, že definované funkce mají pozitivní vliv na predikci sítě a splnily svůj účel. Avšak rozdíl RMSE mezi oběma architekturami byl zanedbatelný. Ve snaze zlepšit výstupy sítě byla vytvořena ještě poslední architektura, která vznikla ze třetí architektury nahrazením skryté plně propojené vrstvy LSTM vrstvou, která je schopna se naučit časové závislosti. S touto sítí se však dosahovalo obdobných výsledků, jako tomu bylo u třetí architektury.

Výsledky všech architektur sítí měly jednu společnou vlastnost. Žádná z vytvořených sítí nedokázala predikovat nečinnost sondy č. 1 v první polovině sezóny vybrané k testování. U experimentů s otevřenou smyčkou zpětné vazby byly predikované hodnoty nízké, ale nikdy nulové, navíc ke snížení došlo u souvislých sekvencí těžby, kterým předcházely dny bez těžby, vždy až po první vysoké predikci. To značí, že předlohová data ve zpětné vazbě měla na tuto anomálii vliv, ale ani tak síť nedokázala predikovat nečinnost dané sondy. Sítě s uzavřenou smyčkou zpětné vazby pak tuto nečinnost zcela ignorovaly. Zkoumání výstupů ze simulátoru (který se v dnešní době využívá pro řízení PZP) na daných datech ukázalo, že simulátor rovněž vypnutí sondy

č. 1 nepredikuje, navíc dosahuje horší hodnoty RMSE na testovací sadě dat než sítě s uzavřenou smyčkou, a to cca o $8\,500\ sm^3$. Vzhledem k faktu, že simulátor predikuje hodnoty pouze na základě stavu kolektoru a nezohledňuje povrchovou technologii, je tedy možné na základě porovnání výstupů simulátoru a sítí konstatovat, že z poskytnutých trénovacích dat se síť nedokáže naučit omezení ze strany povrchové technologie.

V dalším pokračování práce by bylo možné datovou sadu, ze které se síť učí, rozšířit o informace, jaké maximální množství může konkrétní sonda v daný den těžit. Tyto informace by síti poskytly možnost predikce za propojení omezujících vlastností kolektoru a povrchové technologie. Tím by se neuronová síť mohla potenciálně stát významnou částí řízení PZP, protože v dnešní době neexistuje systém, který by obě oblasti dokázal propojit.

Architektura	**Smyčka**	**Průměr RMSE**	**Směrodatná odchylka**
Mělká síť	otevřená	$14\,886,86\ sm^3$	$1\,265,42\ sm^3$
	uzavřená	$25\,537,06\ sm^3$	$657,83\ sm^3$
Validační funkce s oddělenou větví pro zpětnou vazbu	otevřená	$11\,989,61\ sm^3$	$1\,044,47\ sm^3$
	uzavřená	$26\,205,08\ sm^3$	$1\,282,82\ sm^3$
Validační funkce se společnou větví vstupu a zpětné vazby	otevřená	$12\,925,03\ sm^3$	$1\,188,05\ sm^3$
	uzavřená	$25\,788,90\ sm^3$	$822,64\ sm^3$
LSTM vrstva	otevřená	$11\,373,23\ sm^3$	$931,43\ sm^3$
	uzavřená	$26\,286,59\ sm^3$	$917,26\ sm^3$

Tabulka 11.1: Souhrn výsledků dosažených v práci

LITERATURA

[1] ZÁKOPČAN, Marián. HORNICKO-GEOLOGICKÁ FAKULTA, VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA. *Podzemní Zásobníky Plynu*. Hodonín, 2003.

[2] SEVERÝN, Otto a Vladimír ONDERKA. RWE GAS STORAGE S.R.O. *Aplikace numerického modelování v podzemním skladování plynu*. Liberec, 2013.

[3] KROGH, Anders. *What are artificial neural networks?*. Nature biotechnology, 2008, 26.2: 195-197.

[4] M. H. Sazlı *A brief review of feed-forward neural networks*. Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, vol. 50, no. 01, pp. 0-0, Jan. 2006, doi:10.1501/commua1-2_0000000026

[5] *Recurrent Neural Network (RNN). MathWorks [online]*. The MathWorks [cit. 2023-02-13]. Dostupné z: https://www.mathworks.com/discovery/rnn.html

[6] *NARXNET. MathWorks [online]*. The MathWorks [cit. 2023-02-13]. Dostupné z: https://www.mathworks.com/help/deeplearning/ref/narxnet.html

[7] *Deep Learning Toolbox. MathWorks [online]*. The MathWorks [cit. 2023--02-14]. Dostupné z: https://www.mathworks.com/products/deep-learning.html

[8] *Define Shallow Neural Network Architectures. MathWorks [online]*. The MathWorks [cit. 2023-02-14]. Dostupné z: https://www.mathworks.com/help/deeplearning/define-neural-network--architectures.html

[9] *Deep Learning Toolbox: Design, train, and analyze deep learning networks. MathWorks [online]*. The MathWorks [cit. 2023-02-14]. Dostupné z: https://www.mathworks.com/help/deeplearning/index.html

[10] *Network: Create custom shallow neural network [online]*. The MathWorks [cit. 2023-02-14].Dostupné z: https://www.mathworks.com/help/deeplearning/ref/network.html

[11] *Train and Apply Multilayer Shallow Neural Networks [online]*. The MathWorks [cit. 2023-02-16].Dostupné z: https://www.mathworks.com/help/deeplearning/ug/train-and-apply-multilayer-neural-networks.html

[12] *Train [online]*. The MathWorks [cit. 2023-02-16].Dostupné z: https://www.mathworks.com/help/deeplearning/ref/network.train.html

[13] *List of Deep Learning Layers. MathWorks [online]*. The MathWorks [cit. 2023-02-21]. Dostupné z: https://www.mathworks.com/help/deeplearning/ug/list-of-deep-learning-layers.html

[14] *Define Custom Deep Learning Layers. MathWorks [online]*. The MathWorks [cit. 2023-02-21]. Dostupné z: https://www.mathworks.com/help/deeplearning/ug/define-custom-deep-learning-layers.html

[15] *TrainNetwork: Train deep learning neural network. MathWorks [online]*. The MathWorks [cit. 2023-02-23]. Dostupné z: https://www.mathworks.com/help/deeplearning/ref/trainnetwork.html

[16] *TrainingOptions: Options for training deep learning neural network. MathWorks [online]*. The MathWorks [cit. 2023-02-23]. Dostupné z: https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html

[17] *Train Network Using Custom Training Loop. MathWorks [online]*. The MathWorks [cit. 2023-02-24]. Dostupné z: https://www.mathworks.com/help/deeplearning/ug/train-network-using-custom-training-loop.html

[18] *Dlnetwork: Deep learning network for custom training loops. MathWorks [online]*. The MathWorks [cit. 2023-02-24]. Dostupné z: https://www.mathworks.com/help/deeplearning/ref/dlnetwork.html

[19] *Dlarray: Deep learning array for custom training loops. MathWorks [online]*. The MathWorks [cit. 2023-02-24]. Dostupné z: https://www.mathworks.com/help/deeplearning/ref/dlarray.html

[20] *HOLČÍK, Jiří, KOMENDA, Martin (eds.) a kol*. Matematická biologie: e-learningová učebnice [online]. 1. vydání. Brno: Masarykova univerzita, 2015. ISBN 978-80-210-8095-9.

[21] *ACTIVATION FUNCTIONS IN NEURAL NETWORKS. In: International Journal of Engineering Applied Sciences and Technology [online]*. 4. 2020, s. 310-316 [cit. 2023-05-16]. ISSN 2455-2143. Dostupné z: https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf.

SEZNAM PŘÍLOH

1. Data
    (a) data.xslx
    (b) data_modified.xslx
    (c) vysledky_simulatoru.xlsx
    (d) Mat
        i. m01_AllDataTT.mat
        ii. m02_Season.mat
        iii. m03_DataManager.mat
        iv. m04_OptimParamNARXV1.mat
        v. m05_OptimParamNARXV2.mat
2. Zdroje
    (a) Funkce
        i. f01_surfOutput.m
        ii. f02_regressionChart.m
    (b) Objekty
        i. o01_DataManager.m
        ii. o02_MyStandartization.m
        iii. o02b_MyMapMinMax.m
        iv. o03_TrendValidationLayer.m
        v. o04_SumValidationLayer.m
    (c) Pomocné
        i. t01_loadData.m
        ii. t02_selectProduction.m
        iii. t03_dataDependencies.m
        iv. t04_frequencyOfData.m
        v. t05_optimShallowNet.m
        vi. t06_compSimulation.m
    (d) Sítě
        i. ns01_firstExperiment.m
        ii. ns02_addPrefixBeforeSeason.m
        iii. ns03_closeLoop.m
        iv. ns04_closeLoopV2.m
        v. ns05_optimParam.m
        vi. ns06_optimParamV2.m

vii. nd01_ovfOpenLoop.m
viii. nd02_ovfOpenLoopV2.m
ix. nd03_ovfCloseLoop.m
x. nd04_ovfCloseLoopV2.m
xi. nd05_LSTMOpenLoop.m
xii. nd06_FbAsSeqOL.m
xiii. nd07_FbAsSeqOLMTL.m
xiv. nd08_FbAsSeqCL.m