ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

FAKULTA
STROJNÍ

DIPLOMOVÁ PRÁCE

2017

MARTIN
FAYAD

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Fayad** Jméno: **Martin** Osobní číslo: **408549**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídicí techniky**

Studijní program: **Strojní inženýrství**

Studijní obor: **Přístrojová a řídicí technika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Ladící nástroj pro predikci dat energetických budov a detekci neobvyklých stavů neuronovými sítěmi

Název diplomové práce anglicky:

Tunning tool for data prediction of energy buildings and detection of unusual states via neural networks

Pokyny pro vypracování:

1. Proveďte rešerši metod zpracování predikce energetických dat budov a detekce neobvyklých stavů se zaměřením na neuronové sítě.
2. Proveďte rešerši nejběžnějších krokových dávkových učících algoritmů s učitelem pro HONU a MLP (GD, RLS, Lev.-Marquardt, Rprop, konj. Grad., extreme machine learning,...) a zvolte cca 1 metodu krokového učení a 1-2 metody dávkového učení pro vaši aplikaci.
3. Proveďte rešerši a porovnejte vývojová prostředí (jazyky) pro rychlý vývoj vaší aplikace (SW prototyping, včetně možností tvorby GUI, Matlab, Python, C#, ...) a zvolte prostředí a volbu zdůvodněte.
4. Navrhněte a naprogramujte aplikaci s výpočetně efektivní metodou krokového a dávkového učení pro predikci časových řad provozních veličin budov pro HONU (r=1,2,3) a MLP s volitelným počtem neuronů (s jednou skrytou vrstvou)
5. Ověřte funkčnost na umělých datech I reálných datech
6. Zdokumentujte řešení a vyhodnoťte teoretické i praktické poznatky.

Seznam doporučené literatury:

[1] ŠIROKÝ, Jan, Frauke OLDEWURTEL, Jiří CIGLER a Samuel PRÍVARA. Experimental analysis of model predictive control for an energy efficient building heating system. Applied Energy
[2] HUANG, Guang-Bin, Qin-Yu ZHU a Chee-Kheong SIEW. Extreme learning machine: Theory and applications. Neurocomputing [online].

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Ivo Bukovský Ph.D., U12110.3

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

Datum zadání diplomové práce: **19.04.2017** Termín odevzdání diplomové práce: **16.06.2017**

Platnost zadání diplomové práce: _____

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.

19.04.2017
Datum převzetí zadání

Podpis studenta

Souhrn: Cílem a výstupem této práce je návrh a naprogramování aplikace pro predikci dat energetickýcm budov a pro detekci neobvyklých stavů neuronovými sítěmi. Základním požadavkem aplikace je možnost nastavení parametrů neuronové sítě a predikce, výběru neuronového modelu a učícího algoritmu, úpravy dat, volby libovolných vstupních veličin a schopnost detekovat neobvyklé stavy vyskytující se v datech. Práce se zabývá různými přístupy k detekci stavů, které se v rámci dat vymykají běžnému chování avšak ne vždy je lze považovat za neobvyklé. Jsou zde také popsány teoretické poznatky získané při snaze detekovat různé druhy stavů pomocí aplikace. V závěru práce jsou shrnuty možnosti aplikace, výsledky práce a také nastíněn další možný vývoj.

Klíčová slova: Neuronová síť, predikce spotřeby energie, učící algoritmy, aplikace, detekce, neobvyklé stavy

Summary: An aim and an output of this thesis is design and a realization of an application for forecasting data of energy consumption of buildings and for detection of unusual states with artificial neural networks. Main demand of the application is a possibility to set parameters of a neural network and of prediction, to chose neural model and learning algorithm, to preprocess data, adjust inputs to neural network and to detect unusual states appearing in data. The thesis deals with different approaches towards detection of states which appears in data but cannot be always regarded as unusual. A theoretical knowledge received during testing detection on various unusual states with the application are described here. At the end, possibilities of the application, results of the work, and proposals for the future development of the application are summarized.

Keywords: Artifical neural network, prediction, buildings, energy consumption, learning algorithms, application, detection, unusual states

Prohlášení

Prohlašuji, že jsem diplomovou práci s názvem: „Ladící nástroj pro predikci dat energetických budov a detekci neobvyklých stavů neuronovými sítěmi“ vypracoval samostatně pod vedením Doc. Ing. Iva Bukovského, Ph.D., s použitím literatury uvedené na konci mé diplomové práce.

V Praze dne …….. Martin Fayad ……..

Poděkování

Chci poděkovat vedoucímu této práce Doc. Ing. Ivu Bukovskému, Ph. D. za jeho čas, který mi věnoval, pomoc, kterou mi poskytl při vypracovávání této práce a za vše, co jsem se díky němu naučil. Dále děkuji firmě ENERGOCENTRUM Plus, s.r.o. za poskytnutí dat použitých v této práci.

Především však chci poděkovat svým rodičům, kteří mě po celou dobu studia i mimo něj podporovali, kteří mi toho tolik dali a kterým vděčím za svůj život. Bez nich by nebylo nic možné.

Obsah

1 Přehled používaných značení

Značení:

y_n	výstup z neuronu nebo ze sítě.
$\mathbf{u}$	je vektor externích vstupů.
y_t, y_r	naměřené hodnoty, mohou sloužit jako vstup do neuronu
$e_{average}$	průměrná chyba z předchozích chyb
$\mathbf{X^M}$	vstupní matice, jejíž řádky reprezentují vstupní vektory $\mathbf{x^V}$ vstupující do neuronu
$\mathbf{x^V}$	vstupní vektor do neuronu, obsahující vstupní veličiny
$\mathbf{colx}$	dlouhý vektor vypočítaný ze vstupů $\mathbf{x^V}$do neuronu (HONU)
$\mathbf{colX}$	jakobián, vypočítaný ze vstupní matice $\mathbf{X^M}$, jeho řádky jsou tedy $\mathbf{colx}$ (HONU)
f_{HONU}	synaptická operace HONU
$\mathbf{W}$	vektor neurálních vah HONU nebo vektor vah MLP ve skryté vrstvě
$\mathbf{V}$	vektor vah výstupních neuronů MLP
$\boldsymbol{\nu}$	vektor výstupů synaptických operací skrytých neuronů
ϕ	somatická operace
p	horizont predikce

Zkratky:

$RMSE$	Root mean square error
MAE	Mean absolute error
$CoGr$	Coarse graining
GUI	Graphical user interface
MLP	Multi-layer perceptron
$HONU$	Higer order neural units – polynomiální neuronové jednotky
LNU	Linear neural unit – lineární neuronová jednotka
QNU	Quadratic neural unit – kvadratická neuronová jednotka
CNU	Cubic neural unit – kubická neuronová jednotka

BP	Backpropagation – třída učících algoritmů s propagací derivace výstupu sítě
ELM	Extreme learning machine
CG	Conjugate gradient – dávkový algormitmus učení
LM	Levenberg – Marquardt - dávkový algormitmus učení
NGD	Normalized Gradient Descent - krokový algoritmus učení
GD	Gradient Descent – krokový algoritmus učení
RLS	Recursive Least Square – krokový algoritmus učení

2 Úvod

Téma této práce vychází z námětu společnosti ENERGOCENTRUM Plus na detekování neobvyklých stavů při spotřebě energie a navazuje tak na dlouhodobou spolupráci ENERGOCENTRUM Plus s ČVUT v oblasti analýzy provozních dat [1] [2].

Aby bylo možné provádět detekci, je nejprve nutné predikovat spotřebu energie modelem nebo zde neuronovými sítěmi, tj. modelem získaných z reálných dat. Samotná schopnost predikce je též velmi cenná a společně s možností detekce neobvyklých stavů tvoří dle mého názoru velmi žádaný produkt

Prvním cílem této práce je vytvořit neuronový model pro predikci spotřeby energie. Data spotřeby energie pochází z reálných budov a byla mi poskytnuta společností ENERGOCENTRUM Plus. Na základě predikce pak dále detekovat neobvyklé stavy, které se objevují v naměřených datech. Hlavním cílem je pak pro neuronové modely i detekci neobvyklých stavů naprogramovat grafické rozhraní, pomocí kterého by bylo možné ladit jak neuronový model, tak i detekci neobvyklých stavů.

Práce je rozdělena na část teoretickou a část praktickou.

V teoretické části jsou popsány:

- různé modely predikce spotřeby energie se zaměřením na neuronové sítě
- neuronové architektury, které jsou využité v praktické části
- učící algoritmy, z nichž některé jsou využité v praktické části
- vývojová prostředí, z nichž jedno je vybráno pro tvorbu grafického rozhraní

Na konec teoretické části jsem si dovolil přidat vlastní úvahu, pojednávající o charakteru dat, které jsem dostal od ENERGOCENTRUM Plus. V této úvaze se zabývám univerzálními veličinami, na kterých je obecně závislá spotřeba energie budov. Výstupem úvahy je volba veličin, s kterými pracuje můj model predikce spotřeby energie budov .

V praktické části je:

- popsána zvolená metoda predikce
- popsána metoda detekce (učení na chybných datech, chyb v datech)
- popis grafického rozhraní s návodem jak jej používat
- ověření metody predikce a detekce na umělých datech
- dosažené výsledky na datech od ENERGOCENTRUM Plus
- vyhodnocení výsledků a popsány vlastní zkušenosti získané při ladění neuronového modelu a detekce neobvyklých stavů

3 Metody predikce pro spotřebu Energie

Metody zaměřené na predikci spotřeby energie budov lze zařadit do tří hlavních skupin – statistické, inženýrské a metody výpočetní (umělé inteligence). V posledních letech se metody umělé inteligence, jako jsou neuronové sítě, začaly aplikovat pro predikci spotřeby energie. Neuronové sítě mají schopnost postihnout velmi komplexní nelineární vztahy mezi

jejími vstupy a výstupy. Vzhledem k tomu, že proces spotřeby energie má nelineární chování, jsou neuronové sítě vhodné pro predikci spotřeby energie. Navíc je jednoduší vyvinout systém predikce z neuronových sítí, než využívat běžných statistických a inženýrských metod při zachování přesnosti [3] .

Metoda predikce spotřeby energie budovy závisí na každé budově zvlášť. Budovy mají různé parametry, podléhají různým vlivům a proto i metody predikce neuronovými sítěmi se musí přizpůsobit každému problému individuálně. V této kapitole uvedu několik článků s různými metodami predikce spotřeby energie budov se zaměřením na neuronové sítě, které byly ověřeny na reálných datech, nebo které se snažily nahradit příliš komplexní konvenční metody.

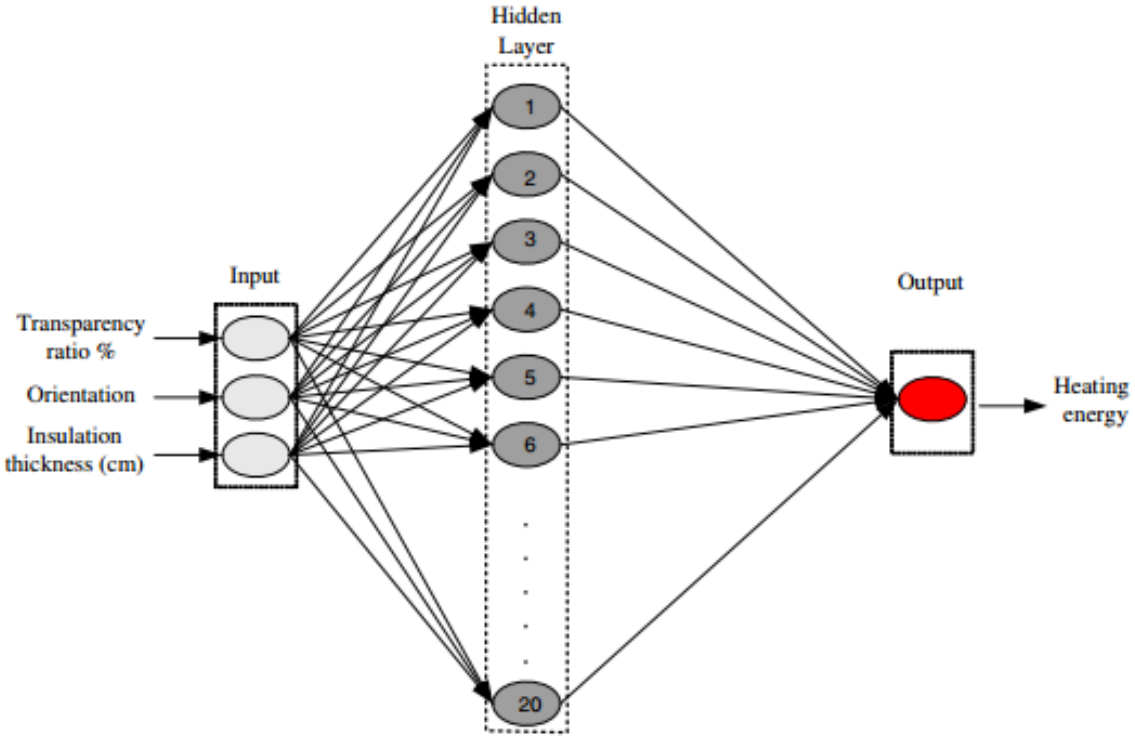
Autor článku [4] použil na model predikce spotřeby energie vytápěním budovy neuronové sítě MLP s učícím algoritmem *backpropagation.* Model se natrénoval na datech spotřeby energie 250 budov s malými i velkými mistnostmi. K dispozici měl velkou škálu vstupních veličin charakterizující budovu – Tab 1.

Plocha oken -[m^2]
Vnější plocha stěn -[m^2]
Plocha podlahy -[m^2]
Střecha/Strop – [1/2]
Typ okna - $[\frac{W}{m^2K}]$
Typ vnitřní zdi - $[\frac{W}{m^2K}]$
Žádaná teplota v dané mistnosti - $[C^o]$

Tab 1: Vstupní veličiny do neuronové sítě

S tímto modelem bylo dosáhnuto v 90% budov přesnosti nad 95% mezi predikovanou a naměřenou hodnotou. Ve zbylých 10% byla přesnost mezi

95% až 90%. V článku [5] autoři predikovali spotřebu energie na vytápění budov bez



Obr. 1: Tří-vrstvá MLP síť s 20 neurony ve skryté vrstvě a s 3 vstupními veličinami – poměr propustnosti, orientace a izolace , obrázek převzat z [5]

uvažování klimatických veličin. Na model predikce byla použita neuronová síť MLP s backpropagation algoritmem. Neuronová síť byla pouze tří-vrstvá se skrytou vrstvou o 20 neuronech (Obr. 1). Vstupem do sítě byly pouze tři veličiny. První byl poměr propustnosti, který procentuálně vyjadřoval poměr zděné plochy k ploše všech vnějších oken. Další veličinou byla orientace budovy, která nabývala hodnot od $0^o - 80^o$ a nakonec tloušťka izolace budovy. Výstupem z neuronu byla hodnota spotřeby energie na vytápění. Dle autorů tento model predikce dosáhl přesnosti mezi 90% až 99% vůči hodnotám z naměřených dat.

V dalším článku [6] autor použil neuronové sítě MLP pro model dlouhodobé predikce, který zahrnuje ve svém modelu počasí a natrénoval tento model na budovách v různých klimatických regionech. K dispozici měl

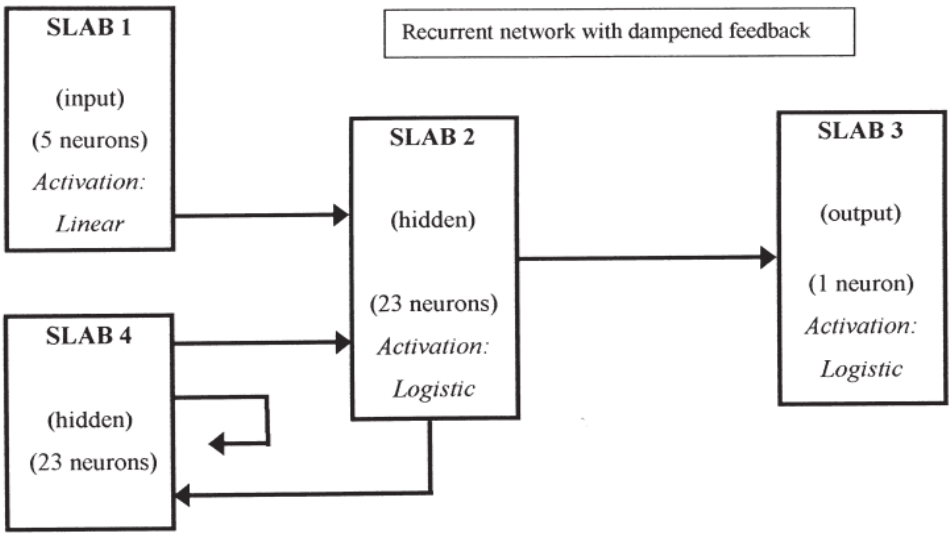
18 veličin charakterizující povrch budovy a 2 další důležité veličiny HDD[1] a CDD[2], které vyjadřují na základě rozdílu průměrné teploty v daném místě a definované základní teploty, zda se v budově bude topit či chladit a určuje, kolik energie se spotřebuje na tuto činnost [7]. Výstupem z neuronu byly dvě veličiny - spotřeba energie na vytápění a na chlazení. Tento model srovnal s výsledky konvenčních metod a simulačních softwarů(DeST, DOE−2, TRNSYS a Energyplus). Model predikce neuronovými sítěmi měl 96% přesnost vůči výsledkům výše zmíněných metod. V tomto ohledu vidí autor článku hlavní výhodu v použití neuronových sítí především ve schopnosti rychle a jednoduše vytvořit model predikce na velmi komplexních problémech.

Autor článku [8] sestavil model predikce spotřeby energie pasivní solární budovy[3], která nepoužívá mechanické ani elektrické zařízení na vytápění. Budova má jednu místnost a nakloněnou střechu. Architekturu neuronové sítě autor zvolil tzv. Jordan-Elmanovu rekurentní síť (Obr. 2), která má 3 vrstvy jako MLP a čtvrtá vrstva slouží jako jakási dlouhodobá paměť skryté vrstvy.

1 Heating degree day

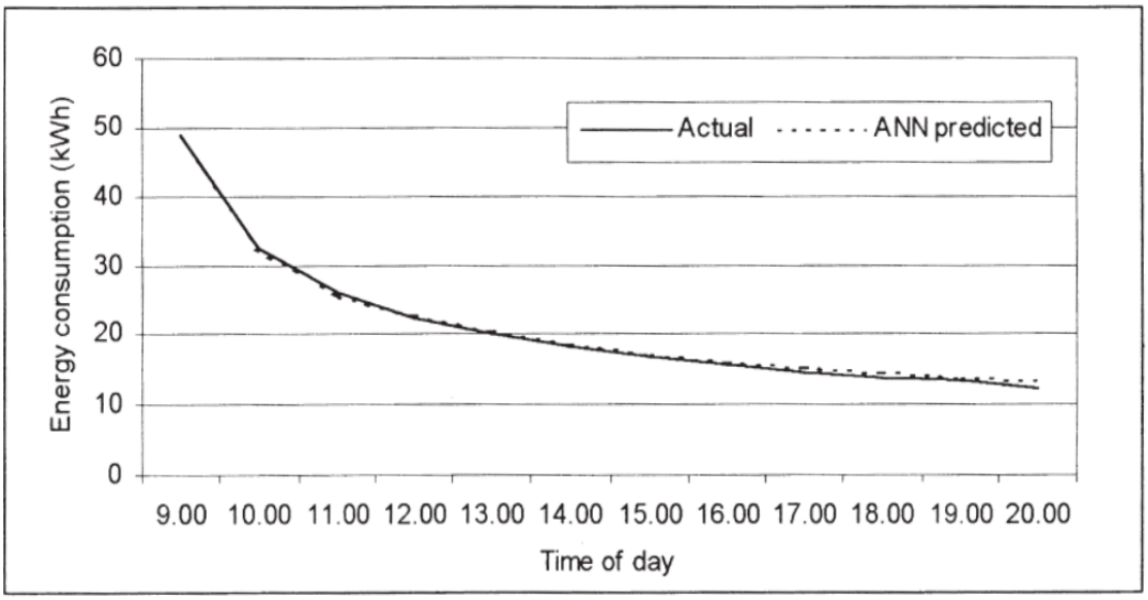
2 Cooling degree day

3 Budova, která v zimě vytápí sluneční energií a v létě udržuje chlad nepřijímáním sluneční energie



Obr. 2: Využití Jordan-Elmanovy rekurentní síťě pro predikci spotřeby pasivní solární budovy, převzato z [8]

Účel této čtvrté vrstvy je, aby si neuronová síť pamatovala předchozí vstupy. Do neuronové sítě vstupují čtyři parametry – sezóna(léto,zima), izolace(1 nebo 0), tloušťka zdi, konstantnost součinetele přestupu tepla(1 nebo 0) a čas dne. Výstup z neuronu je spotřeba energie budovy. Učící algoritmus byl zvolen backpropagation. Autor článku měl k dispozici data ze simulace spotřeby energie této budovy na základě matematického – fyzikálního modelu. Na tyto data se model predikce natrénoval a ozkoušel se na naprosto nových datech. Přesnost toho velmi jednoduchého modelu byla téměř shodná s matematicko – fyzikálním modelem (Obr. 3).



Obr. 3: Srovnání matematicko - fyzikálního modelu s neuronovou sítí, převzato z [8]

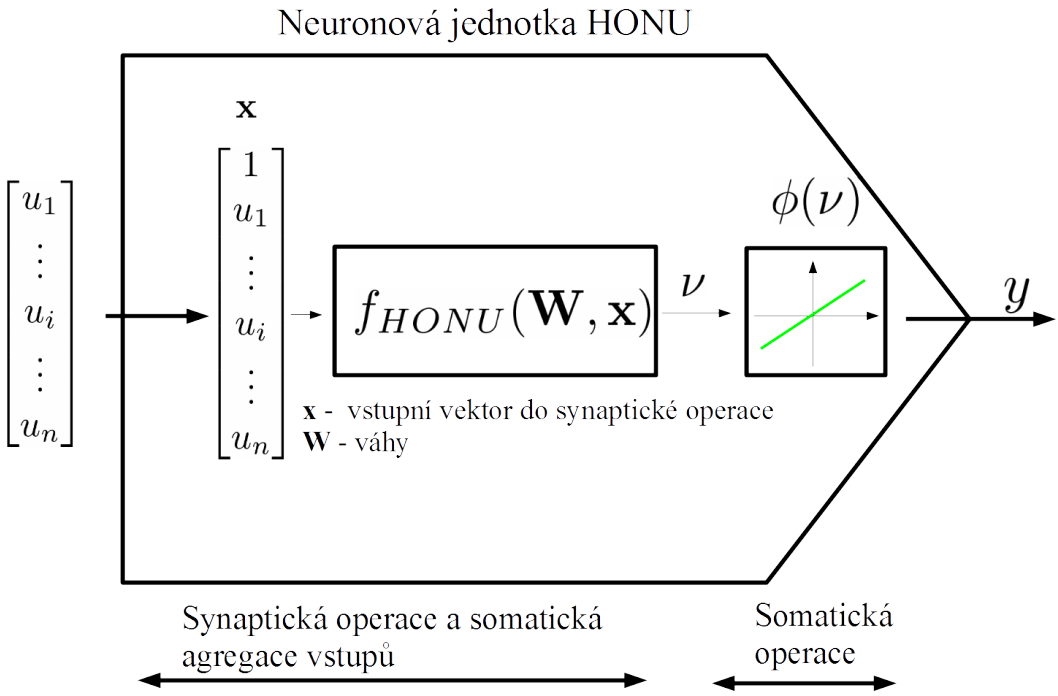
Celkově vzato patří mezi nejoblíbenější neuronové modely predikce spotřeby energie neuronové sítě typu MLP, které využívají učící algoritmy Backpropagation.

4 Neuronové architektury

V této práci je použito pro predikci spotřeby energie několik různých neuronových architektur spadající do tří skupin. První skupinou je klasická neuronová síť typu MLP, druhou je lineární neuronová jednotka (LNU) a třetí jsou HONU s druhým a třetím stupněm polynomu, tj. kvadratické (QNU) a kubické (CNU) neuronové jednotky [9]. LNU lze považovat za HONU prvního stupně a proto jej zahrnu k HONU. Dále uvedený text předpokládá u čtenáře obecnou znalost principu fungování neuronových sítí.

4.1 Higher order neural units (HONU)

HONU bývají také označovány jako HONNU (Higher Order Nonlinear Neural Units - nelineární neuronové jednotky s nelinearitami vyšších řádů [7]. Nelineární neuronové modely HONU jsou v přímé analogii k Taylorovu polynomiálnímu rozvoji. HONU má volitelný řád polynomu a proto lze



Obr. 4: Schéma neuronu HONU, překresleno z [35]

nastavovat kvalitu nelineární aproximace. Dnešní hardware výpočetně umožňuje zvládat strojové učení HONU až do polynomu 3. stupně [10]. S vyššími stupni polynomu a množstvím vstupů roste složitost výpočtu a tedy i časová náročnost. V této práci jsou použity modely HONU LNU, QNU a CNU. LNU má na rozdíl od QNU a CNU lineární synaptickou operaci. Všechny tři mnou použité modely však lze vystihnout Obr. 4.

Rozdíl mezi modely spočívá v agregační funkci f_{HONU}.

Pro LNU je agregační funkce lineární a rovna

$$f_{HONU} = \sum_{i=0}^{n} x_i \cdot w_{i,} \tag{4.1}$$

pro QNU je nelineární a rovna

$$f_{HONU} = \sum_{i=0}^{n} \sum_{j=i}^{n} x_i \cdot x_j \cdot w_{i,j,} \tag{4.2}$$

a pro CNU je též nelineární a rovna

$$f_{HONU} = \sum_{i=0}^{n} \sum_{j=i}^{n} \sum_{k=j}^{n} x_i \cdot x_j \cdot x_k \cdot w_{i,j,k} \tag{4.3}$$

4.1.1 Modifikace HONU

V této podkapitole popíšu dle [11] metodu, se kterou lze implementovat neuronové jednotky HONU jednodušeji a výpočetně rychlejší. Jednodušeji proto, že v této metodě převádíme N-dimensionální matice do vektorů a pracujeme tedy jen s vektory. Rychlejší proto, že jednak při výpočtu nemusí HW pracovat s N-dimensionálními maticemi, které při větších rozměrech ubírají mnoho operační paměti a jednak je nahrazeno násobení jednotlivých prvku matic jednou operací – maticovým násobením.

U neuronových modelů HONU má synaptická operace tvar

$$f_{HONU}(\mathbf{x}, \mathbf{W}),$$

kde

$\mathbf{x}$ je vstupní vektor,

$\mathbf{W}$ je matice vah.

Z hlediska výpočetní rychlosti je vhodné provést synaptickou operaci maticově. Například neuronová jednotka QNU, tedy polynom 2. stupně, má výstup vyjádřen jako

$$y_n = \sum_{i=0}^{n} \sum_{j=i}^{n} x_i \cdot x_j \cdot w_{i,j,} \tag{4.4}$$

a kde matice vah w je 2D a má tento tvar

$$\mathbf{W} = \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{1,n} \\ 0 & w_{1,1} & \cdots & w_{2,n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & w_{n,n} \end{bmatrix}. \tag{4.5}$$

Z (4.5) je využitá jen horní trojúhelníková matice, tedy jen polovina, neboť prvky matice, které jsou nulové, neovlivní výstup a jsou tedy zbytečné. V případě neuronové jednotky s polynomem 3. stupně bude využito při výpočtu ještě méně.

Aby se předešlo náročnému výpočtu neuronového výstupu sumací součinu prvků vstupního vektoru a prvků v matici vah (4.4), převede se vstupní vektor a matice vah do dvou tzv. *dlouhých vektorů*

$$row(\mathbf{W}) = row_{\mathbf{W}}, \tag{4.6}$$

$$row^r(\mathbf{x}) = row_{\mathbf{x}}, \tag{4.7}$$

kde r je řád polynomu.

Má-li vstupní vektor n vstupů, pak délka l *dlouhého vektoru* $row_{\mathbf{W}}$ *a* $row_{\mathbf{x}}$ vychází ze sumace (synaptické operace) (4.4) a je rovna pro
polynom 1. stupně (LNU)

$$l^{r=1} = n, \tag{4.8}$$

polynom 2. stupně (QNU)

$$l^{r=2} = \frac{n \cdot (n+1)}{2}, \tag{4.9}$$

a polynom 3. stupně (CNU)

$$l^{r=3} = \frac{n \cdot (n+1) \cdot (n+2)}{6}. \tag{4.10}$$

N-Dimensionální matici vah tedy lze přepsat do 1D vektoru o délce l^r.
Nový vektor vstupů bude mít tvar pro QNU

$$row^{r=2}(\mathbf{x}) = [x_0^2 \ x_0x_1 \ \cdots x_ix_j \ x_n^2], \tag{4.11}$$

kde budou jednotlivé součiny vstupního vektoru seřazeny stejným pravidlem, ze kterého vyplývá sumace z (4.4).

Neuronový model HONU lze tedy na základě předešlé implementace vyjádřit obecně pro libovolný stupeň polynomu takto

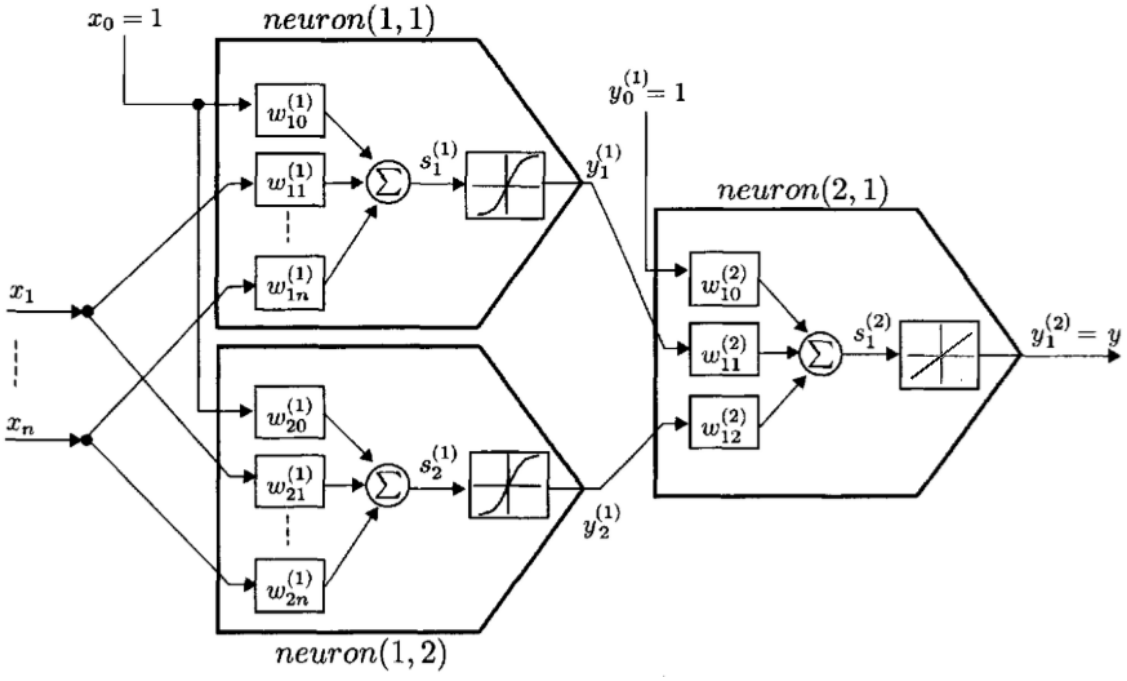
$$y_n = row^r(\mathbf{x}) \cdot row(\mathbf{W})^T = row^r(\mathbf{x}) \cdot col(\mathbf{W}), \tag{4.12}$$

kde $col()$ analogicky představuje operaci vytvoření dlouhého sloupcového vektoru.

Tuto modifikaci jsem implementoval v aplikaci pro rychlý výpočet HONU.
V knihovně neural_network.py jsou funkce, pomocí kterých lze používat tuto implementaci a v souboru neural_network_functions.py jsou vzorové příklady použití neuronových sítí s touto implementací, viz. příloha (kap. 14).

4.2 Více-vrstvá neuronová síť (MLP)

MLP je model dopředné neuronové sítě [12]. Síť se skládá ze tří a více vrstev – vstupní, skrytých a výstupní. Vstupní vrstva reprezentuje vstupní vektor do sítě. Vstupy prostupují do sítě vždy dopředu přes všechny vrstvy a proto se síť nazývá dopředná. Skryté vrstvy obsahují neurony, které mají strukturu jako LNU neuron s nelineární aktivační funkcí. Neurony ve výstupní vrstvě mají obvykle lineární aktivační funkci (Obr. 5).



Obr. 5: Tří-vrstvá neuronová síť MLP, převzato z [12]

Nejčastěji používanou nelineární výstupní funkcí ze skrytých vrstev u MLP je sigmoidální aktivační funkce. Její důležitá vlastnost je hladkost v celém intervalu, tzn. je diferencovatelná. Pro normalizovaná data v rozsahu $(-1, 1)$ je vhodné upravit aktivační funkci na tento rozsah a tedy

$$\phi(\boldsymbol{\nu}) = \frac{2}{1 + e^{-\boldsymbol{\nu}}} - 1. \tag{4.13}$$

Ve skryté vrstvě neurony agregují vážené vstupy

$$\boldsymbol{\nu} = \mathbf{W} \cdot \mathbf{x}, \tag{4.14}$$

a vrací výstup přes aktivační funkci

$$\phi(\boldsymbol{\nu}) = \phi(\mathbf{W} \cdot \mathbf{x}). \tag{4.15}$$

Má-li neuron ve výstupní vrstvě lineární aktivační funkci, pak jsou vstupem do výstupního neuronu výstupy z neuronů skryté vrstvy a tedy výstup výstupní vrstvy je roven

$$y = \mathbf{V} \cdot \phi(\boldsymbol{\nu}) = \mathbf{V} \cdot \phi(\mathbf{W} \cdot \mathbf{x}), \tag{4.16}$$

kde $\mathbf{V}$(v Obr. 5 $w^{(2)}$) je vektor vah výstupního neuronu a $\mathbf{W}$ (v Obr. 5 $w^{(1)}$) je matice vah neuronů ve skryté vrstvě.

5 Učící algoritmy

Abychom mohli používat neuronové modely, je potřeba je nejdříve naučit závislosti vstupů na výstupu pomocí trénovacích dat. V této kapitole je představeno několik nejznámějších učících algoritmů s učitelem.

5.1 Backpropagation

Rozsáhlou skupinou učících algoritmů je tzv. Backpropagation (BP). Poprvé se BP objevil v 70. letech minulého století, ale jeho význam byl doceněn až roku 1986, kdy byl vydán článek, popisující neuronové sítě s algoritmy BP, které se učily rychleji, než algoritmy předešlé. V té době se jednalo o přelom, neboť bylo možné neuronovými sítěmi řešit problémy, které do té doby byly nevyřešitelné. Dnes jsou algoritmy BP základem při učení i hlubokých neuronových sítí [13].

Učení algoritmů BP spočívá ve zpětném šíření chyby. Na výstupu neuronové sítě se sleduje chyba e (5.1) mezi hodnotou výstupu z neuronu y_n a reálnou (naměřenou) hodnotou y_t (index t jako "target").

$$e = y_t - y_n. \tag{5.1}$$

Učení vychází z tzv. chybové funkce, která vyplývá z (5.1)

$$E = C(y_t - y_n)^2, \tag{5.2}$$

Chybová funkce je závislá na reálných datech, vstupním vektoru do neuronu a neurálních váhách. Pro různé hodnoty vah má chybová funkce různou hodnotu. Hledají se tedy váhy s takovými hodnotami, při kterých je chybová funkce nejmenší, tedy v globálním minimu. Aby hledání vah bylo deterministické, využívá BP metodu klesajícího gradientu pro nalezení minima chybové funkce E

Na základě gradientu chybové funkce

$$\nabla E = \frac{\partial E}{\partial w} \tag{5.3}$$

se přizpůsobují váhy. Adaptaci vah lze vypočítat v každém výpočetním čase podle zjednodušeného klasického schématu (5.4)(5.5) dle [14]

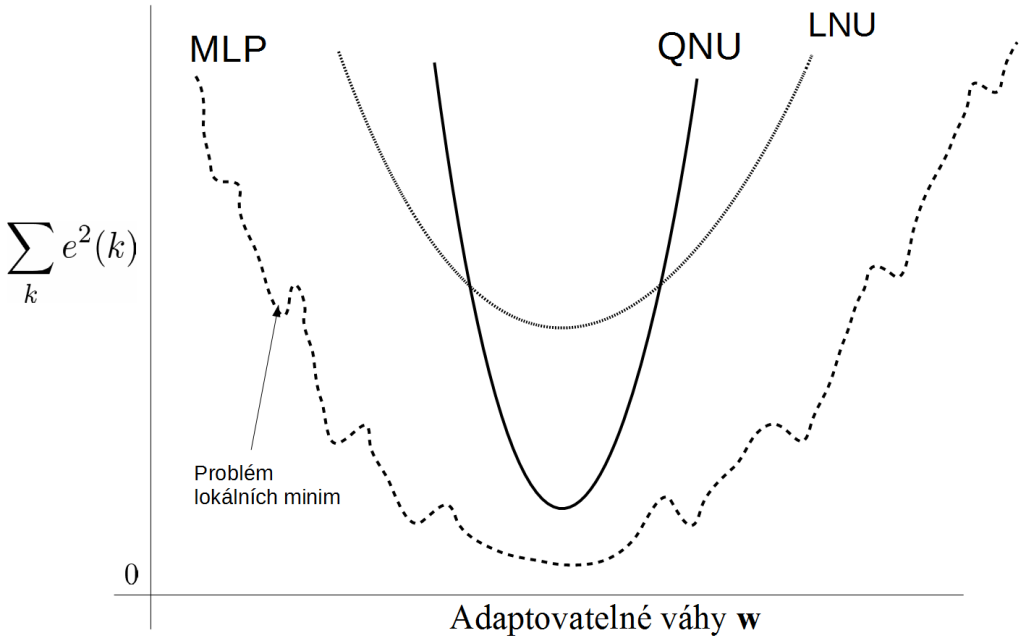
$$\Delta w = -\frac{1}{2}\mu\frac{\partial e^2}{\partial w} \tag{5.4}$$

$$w = w + \Delta w \tag{5.5}$$

kde C je konstanta a obvykle se volí

$$C = \frac{1}{2} \text{ (viz. 5.1.1).}$$

Obrázek (Obr. 6) názorně vystihuje chování různých neuronových architektur vzhledem k chybové funkci. Modely HONU (LNU, QNU, CNU) se dokáží naučit velmi rychle a netrpí lokálními minimy chybové funkce. Sítě MLP, které využívají algoritmů BP, se dokáží naučit daný problém velmi přesně, avšak chybová funkce trpí lokálními minimy.



Obr. 6 Chybová funkce pro odlišné neuronové modely dle [36]

5.1.1 Gradient Descent (GD)

GD je jeden z nejoblíbenějších algoritmů pro optimalizaci obecně a zvláště pak pro neuronové sítě. GD je jádrem BP a tedy i všechny algoritmy BP výchazi z GD. Rovnici adaptaci jednotlivých vah i metodou GD získáme z (5.1),(5.3),(5.2),(5.5) a lze ji zapsat takto

$$w_i(k+1) = w_i(k) - \mu \cdot \frac{\partial E}{\partial w_i}. \tag{5.6}$$

a po dosazení chybové funkce

$$w_i(k+1) = w_i(k) - \frac{1}{2} \cdot \mu \cdot \frac{\partial e^2}{\partial w_i} \tag{5.7}$$

Konstanta $\frac{1}{2}$ slouží jako „kosmetická" úprava, která se vykrátí po derivaci kvadratického členu.

Nevýhodou GD je, že v určitých případech trpí lokálními minimy. Výhodou naopak je velmi jednoduchá implementace a rychlost výpočtu. Zvolíme-li vysokou rychlost učení μ, pak bude chybová funkce E více kolísat a GD se tak částečně ubrání lokálním minim. Na druhou stranu větší kolísavost způsobí „přestřelování" a chybová funkce se tak nikdy nedostane do globálního minima. Řešením pak je Normalized Gradient Descent (NGD), který normalizuje rychlost učení v každém kroku.

5.1.2 Recursive Least Squares (RLS)

Algoritmus RLS byl vyvinut Gaussem roku 1821, ale byl přehlédnut až do 50. let minulého století. RLS je základní algoritmus třídy rekurzivních algoritmů. RLS vychází z teorie Kalmanovy filtrace a z metody nejmenších čtverců. RLS algoritmy jsou známy pro výborné výsledky v časově proměnných prostředích, nicméně za cenu celkem komplexního výpočtu a problémů se stabilitou [15].

Pro názornost implementace RLS agoritmu pro adaptaci vah aplikuji RLS algoritmus na neuronové architektuře HONU. Přírůstek vah je dán rovnicí

$$\Delta \mathbf{w}(k) = e(k) \cdot \mathbf{j}(k) \cdot \mathbf{R}^{-1}(k). \tag{5.8}$$

Z rovnice (5.1) vypočtu chybu $e(k)$ v každém kroku k. Vstupní vektor do neuronu $\mathbf{j}(k)$ je v HONU reprezentován jako $\mathbf{colx}$ a $\mathbf{R}$ je matice zesílení [13].

Při implementování RLS algoritmu [9], nejdříve získáme matici zesílení $\mathbf{R}$ z počátečních podmínek, tedy

$$\mathbf{R}^{-1}(k=0) = \frac{1}{\delta}\mathbf{I} \tag{5.9}$$

kde δ je malé kladné číslo, velmi blízko, avšak menší než 1.

Iterace začíná v kroku $k = 1$ výpočtem matice zesílení

$$\mathbf{R}^{-1}(k) = \frac{1}{\mu}(\mathbf{R}^{-1}(k-1) - \frac{\mathbf{R}^{-1}(k-1)\mathbf{j}^T\mathbf{j}\mathbf{R}^{-1}(k-1)}{\mu + \mathbf{j}\mathbf{R}^{-1}(k-1)\mathbf{j}^T}), \tag{5.10}$$

dále se vypočte chyba $e(k)$ a společně s maticí zesílení $\mathbf{R}^{-1}(k)$ se dosadí do rovnice (5.8)

a nakonec se adaptují váhy podle (5.5). Výsledná rovnice adaptace vah při iterování je rovna

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{1}{\mu}(\mathbf{R}^{-1}(k-1) - \frac{\mathbf{R}^{-1}(k-1)\mathbf{j}^T\mathbf{j}\mathbf{R}^{-1}(k-1)}{\mu + \mathbf{j}\mathbf{R}^{-1}(k-1)\mathbf{j}^T}) \tag{5.11}$$

5.1.3 Levenberg – Marquardt (LM)

Algoritmus LM byl vyvinut nezávisle na sobě Levenbergem a Marquardte. LM je iterační metoda, která hledá minimum chybové funkce. LM se stala standardní nelineární metodou nejmenších čtverců a dá se o ní smýšlet jako o kombinaci Gradient Descentu a Gauss – Newtonovy metody. Ze začátku, kdy je řešení daleko, LM postupuje jako GD, sice pomalu, ale s garancí konvergence k minimu. Když je řešení blízko, stane se LM Gauss-Newtonovou metodou a konvergence k minimu se značně urychlí [16].

LM patří do skupiny dávkových učení a váhy se tedy adaptují vždy přes celá data.

LM byl navržen tak, aby dosáhl rychlosti učení druhého řádu [17] bez nutnosti počítat Hessovu matici. Hessova matice je čtvercová matice druhých

parciálních derivací skalární funkce [18]. Budeme-li uvažovat hledání minima chybové funkce jako sumy kvadrátu chyb,

$$E = \frac{1}{2}\sum_{k=0}^{N} e^2(k), \tag{5.12}$$

lze pak aproximovat Hessovu matici jako

$$\mathbf{H} = \mathbf{J}^T \cdot \mathbf{J}, \tag{5.13}$$

gradient se spočte jako

$$\mathbf{g} = \mathbf{J}^T \cdot \mathbf{e}, \tag{5.14}$$

kde $\mathbf{e}$ je vektor chyb neuronové sítě a $\mathbf{J}$ je matice Jakobiánu, která obsahuje první derivace chyb neuronové sítě k neurálním váhám a bias (vychýlení, slouží k vymanění se z lokálních minim).

LM využívá aproximaci Hessovy matice pro adaptaci vah takto

$$\mathbf{w} = \mathbf{w} - [\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{e}, \tag{5.15}$$

kde $\mathbf{I}$ je jednotková matice.

Je – li rychlost učení $\mu = 0$, pak se LM chová jako Newton – Gaussova metoda. Je – li μ veliké, pak se chová jako Gradient Descent s malým krokem. Po každém úspěšném kroku se hodnota μ zmenší, chybová funkce se blíží k minimu a v určitém momentu začne převažovat Newton – Gaussova metoda [19].

LM je rychlý a stabilně konvergující algoritmus, který se zdá být jeden z nejrychlejších pro středně veliké neuronové sítě, tedy pro sítě s váhami o počtu do několika set.

5.1.4 Resilient Propagation (RProp)

Resilient Propagation je heuristický učící algoritmus. Byl vytvořen Martinem Riedmillerem and Heinrichem Braunem v roce 1992. Podstatou RProp je práce pouze se směrem gradientu chybové funkce a nikoliv s její velikosti. Dalším rozdílem je individuální přístup ke každé váze. Uvedu zde RProp algoritmus přizpůsoben pro dávkové učení dle [9].

Adaptace vah je dána rovnicí

$$\mathbf{w}(\epsilon+1) = \mathbf{w}(\epsilon) + \Delta\mathbf{w}(\epsilon) \tag{5.16}$$

kde index ϵ je epocha učení a

$$E(\epsilon) = \sum_k e^2(k) \tag{5.17}$$

je chybová funkce.

Pro určení přírůstku vah je nejprve nutné určit směr gradientu chybové funkce na základě znamének vektoru parciálních derivací

$$\nabla\mathbf{E}(\epsilon) = 2\sum_k \frac{\partial e^2(k)}{\partial \mathbf{w}} = \sum_k e(k)\frac{\partial y_n(k)}{\partial \mathbf{w}}. \tag{5.18}$$

Dále se vyhodnotí každá váha individuálně v jedné epoše ϵ

$$\Delta(\epsilon) = \begin{cases} \eta^+ \cdot \Delta(\epsilon-1) & if \quad \nabla\mathbf{E}(\epsilon-1)\cdot\nabla\mathbf{E}(\epsilon) > 0 \\ \eta^- \cdot \Delta(\epsilon-1) & if \quad \nabla\mathbf{E}(\epsilon-1)\cdot\nabla\mathbf{E}(\epsilon) < 0, \\ \Delta(\epsilon-1) & if \quad \nabla\mathbf{E}(\epsilon-1)\cdot\nabla\mathbf{E}(\epsilon) = 0 \end{cases} \tag{5.19}$$

a podle směru gradientu se vynásobí faktorem η, kde $0 < \eta^- < 1 < \eta^+$. Nakonec se rozhodne o směru přírůstku

$$\Delta w(\epsilon) = \begin{cases} -\Delta(\epsilon) & if \quad \nabla\mathbf{E}(\epsilon) > 0 \\ \Delta(\epsilon) & if \quad \nabla\mathbf{E}(\epsilon) < 0 \\ 0 & if \quad \nabla\mathbf{E}(\epsilon) = 0 \end{cases}. \tag{5.20}$$

Přírůstek váhy z rovnice (5.20) dosadím do (5.16) a získám tak rovnici adaptaci vah pro RProp algoritmus.

5.1.5 Konjugovaný Gradient (CG)

CG byl vyvinut nezávisle E. Stiefelem a M.R.Hestenesem. CG je iterační algoritmus pro numerické řešení systému určitých lineárních rovnic, konkrétně těch, jejichž matice je symetrická a positivně definitní. Vzhledem k této práci lze mluvit o algoritmu dávkového učení [20],[21].

Předností CG je rychlé řešení systému rovnic

$$\mathbf{A} = \mathbf{x}\cdot\mathbf{B}, \tag{5.21}$$

kde $\mathbf{x}$ je neznámý vektor, $\mathbf{B}$ je známý vektor a $\mathbf{A}$ je známá čtvercová, symetrická, positivně definitní matice. Pro názornost převedu problém na řešení adaptace vah neuronové architektury HONU dle [9], z (5.21) získám rovnici

$$\mathbf{b} - \mathbf{A} \cdot \mathbf{w}(\epsilon) = 0, \tag{5.22}$$

která se rozepíše na

$$\mathbf{colX}^T \cdot (\mathbf{y} - \mathbf{colX} \cdot \mathbf{w}(\epsilon)) = 0, \tag{5.23}$$

kde matice $\mathbf{colX} = \mathbf{J}$ (Jakobián).

Vektor $\mathbf{b}$ je roven součinu vstupního vektoru do neuronu (Jakobián, nebo-li $\mathbf{colX}$) a vektoru reálných naměřených hodnot, tedy

$$\mathbf{b} = \mathbf{J}^T \cdot \mathbf{y_t} \tag{5.24}$$

Získám reálnou symetrickou a pozitivně definitní matici

$$\mathbf{A} = \mathbf{J}^T \cdot \mathbf{J}. \tag{5.25}$$

Nyní je potřeba stanovit počáteční podmínky pro první epochu $\epsilon = 0$, tedy

$$\mathbf{r_e}(\epsilon = 0) = \mathbf{b} - \mathbf{A} \cdot \mathbf{w}(\epsilon = 0) \text{ a} \tag{5.26}$$

$$\mathbf{p}(\epsilon = 0) = \mathbf{r_e}(\epsilon = 0) \tag{5.27}$$

Nyní máme vše pro to, abychom mohli začít iteraci. Iterace probíhá tak dlouho, dokud je poměr chyby nové epochy a minulé epochy větší než konstanta c, tedy

$$\beta(\epsilon) = \frac{\mathbf{r_e}^T(\epsilon + 1) \cdot \mathbf{r_e}(\epsilon + 1)}{\mathbf{r_e}^T(\epsilon) \cdot \mathbf{r_e}(\epsilon)} > c \tag{5.28}$$

Cyklus iterace probíhá v tomto pořadí: (5.29),(5.30),(5.31),(5.28),(5.32).

$$\alpha(\epsilon) = \frac{\mathbf{r_e}^T(\epsilon) \cdot \mathbf{r_e}(\epsilon)}{\mathbf{p}^T(\epsilon) \cdot \mathbf{A} \cdot \mathbf{p}(\epsilon)}, \tag{5.29}$$

$$\mathbf{w}(\epsilon + 1) = \mathbf{w}(\epsilon) + \alpha(\epsilon) \cdot \mathbf{p}(\epsilon), \tag{5.30}$$

$$\mathbf{r_e}(\epsilon + 1) = \mathbf{r_e}(\epsilon) - \alpha(\epsilon) \cdot \mathbf{A} \cdot \mathbf{p}(\epsilon), \tag{5.31}$$

$$\mathbf{p}(\epsilon+1) = \mathbf{r_e}(\epsilon+1) + \beta(\epsilon) \cdot \mathbf{p}(\epsilon). \tag{5.32}$$

Splňuje-li poměr chyb β požadavek, iterace končí. CG gradient je velmi rychlý a proto se hodí i na rozsáhlá data. Jeho výhoda je však i nevýhoda, neboť má tendenci se po několika epochách velmi rychle na datech přeučit.

5.2 Extreme Learning Machine (ELM)

ELM algoritmus je nový učící algoritmus pro dopředné neuronové sítě s jednou skrytou vrstvou (single-layer feedforward neural networks – SLFNs) a již nepatří do skupiny Backpropagation [22]. Rychlost učení neuronových sítí je obecně mnohem pomalejší než je v mnohých oblastech využití vyžadováno a příčinou by mohly být dva důvody. První je pomalost gradientně založených algoritmů a druhý je ladění všech parametrů iteračně právě pomocí těchto algoritmů [23].

Metoda ELM dle jejího autora [24] ,[25]. Mám model neuronu

$$y_n = \mathbf{W_2} \cdot g(\mathbf{W_1 x}), \tag{5.33}$$

kde

$\mathbf{W_1}$ je matice vah mezi vstupní a skrytou vrstvou neuronové sítě,

g je některá z aktivačních funkcí,

$\mathbf{x}$ je vstupem do neuronové sítě,

$\mathbf{W_2}$ je matice vah mezi skrytou vrstvou a výstupní vrstvou.

Váhy $\mathbf{W_1}$ se zvolí náhodně a jako takové budou konstantní. Váhy $\mathbf{W_2}$ se též navolí náhodně, ale budou se adaptovat. Dále se spočítá matice H, která reprezentuje výstup ze skryté vrstvy, tedy

$$\mathbf{H} = g(\mathbf{W_1 x}). \tag{5.34}$$

Spočítá se Moore-Penroseova pseudo-inverzní matice

$$\mathbf{H} \rightarrow \mathbf{H}^{\dagger}, \tag{5.35}$$

vypočte se výstup z neuronové sítě

$$y_n = \mathbf{W_2 H} \tag{5.36}$$

a nakonec se přepočtou hodnoty matice vah $\mathbf{W_2}$

$$\mathbf{W_2} = \mathbf{H}^{\dagger} y_n \tag{5.37}$$

Dle autora [24],[26] se ELM učí extrémně rychle, chybová funkce netrpí lokálními minimy ani nevhodnou rychlostí učení a obecně jako model je velmi jednoduchý a přesto účinný.

Z výše uvedených algoritmů jsem si pro praktickou část vybral algoritmus krokového učení GD a z dávkového učení LM a CG (pouze pro HONU). Důvod, proč jsem si vybral optimalizační algoritmy Backpropagation je kromě jejich dobrých výsledků i ten, že jejich porozuměním a jejich aplikací získám nové dovednosti v oblasti optimalizace, které se mi budou hodit nejen při práci s neuronovými sítěmi.

6 Vývojové prostředí

V této kapitole uvedu několik příkladů vývojového prostředí, ve kterém by bylo možné zpracovat zadání diplomové práce. Obsahem bude stručný popis prostředí a jejich klady a zápory ve vztahu k zadání.

6.1 Programovací jazyky a SW prototyping

Software (dále jen SW) prototyping je vytváření testovacích a nekompletních SW programů za účelem testování jednotlivých částí budoucího finálního SW a získání zpětné vazby při projektování. Díky SW prototypingu lze předejít mnoha překážkám a nesnázím, které jsou spojeny s tvorbou komerčního produktu [27]. SW prototyping lze dát do přesné analogie s tvorbou prototypů ve strojírenství či leteckém průmyslu.

V praxi lze pro SW prototypy používat jiný programovací jazyk, než pro finální produkt. Důvod je ten, že je potřeba realizovat či testovat nápady rychle, což v nižších programovacích jazycích, které se běžně používají na rozsáhle SW projekty, nejde. Každý z výše uvedených programovacích jazyků

má svou oblast, ve které se může uplatnit jako prototypovací nástroj. Největší oblast však pokrývá Python. Jeho syntaxe je velmi expresivní a díky jeho silné komunitě zasahuje jeho pole působnosti do mnoha oblastí.

6.1.1 Matlab

Matlab (matrix laboratory) je interaktivní programové prostředí s orientací na technické (numerické) výpočty. Byl vyvinut společností MathWorks v roce 1984 a je standardním nástrojem pro vědecké, výzkumné a technické účely [28],[29] .

Matlab jako celek se skládá z pěti částí:

1. Matlab jazyk – je skriptovací jazyk, s velmi jednoduchou syntaxí. Podporuje tvoření funkcí i objektů, avšak nedá se o něm říct, že by se jednalo o plně objektově orientovaný jazyk. Matlab jazyk je navržen tak, aby se dal osvojit i bez širších znalostí programování.
2. Pracovní prostředí – obsahuje skript s kódem, konzoly pro příkazy, okno s proměnnými a nástroje pro práci s daty.
3. Grafický systém – je napojen na skriptovací jazyk. Jednoduchými příkazy lze vizualizovat data, vytvářet jednoduchou grafiku pro animace či provádět zpracování obrazu. Součástí grafického systému je též tvorba grafického rozhraní
4. Knihovna funkcí – obsahuje velké množství algoritmů a nástrojů pro technické výpočty.
5. Matlab API – je knihovna, která umožňuje psaní programů v C a ve Fortranu, které mohou interagovat s Matlabem

Výhody:

- Maticové výpočty
- Profesionální podpora
- Mnoho nástrojů pro technické výpočty
- Intuitivní vykreslování grafů

- Modelování dynamických systémů, počítačová simulace (Simulink)
- Velmi rychlý vývoj aplikací - tvorba grafického rozhraní

Nevýhody

- Mimo maticové výpočty je pomalý
- Neúsporná práce s pamětí
- Způsob, jakým se tvoří funkce
- Matlab jazyk není plně objektově orientovaný jazyk
- Velmi malé možnosti grafického rozhraní
- Není open-source
- Jednostranné zaměření (vědecko - technické)

6.1.2 Julia

Julia je velmi nový (2012) skriptovací jazyk, který vznikl zejména pro vědecké výpočty a je open-source. Snadno umožňuje paralelní distribuované výpočty a dokáže volat funkce psané v jazyce C, Fotran nebo v Pythonu. Jádro jazyka Julia je implementováno v C a C++. Co je na Julii atraktivní je, že je to jazyk vysokoúrovňový (má podobnou syntaxi jako Matlab) a současně je jeho rychlost srovnatelná s jazykem C [30].

Julia má mnoho společného s Matlabem. Narozdíl však od Matlabu je velmi nová a proto zatím nemá silné zázemí ve vědeckotechnické komunitě. Julia je open-source a proto je závislá právě na velikosti komunity. Z tohoto důvodu chybí Julii mnoho nástrojů, algoritmů a jiných prvků, které komerční Matlab dělají ve vědeckotechnické oblasti jedinečným.

Výhody:

- Velmi podobná syntaxe jako má Matlab
- Nástroje pro technické výpočty
- Skriptovací jazyk se srovnatelnou rychlostí s jazykem C

- Interakce s jazyky C, Fortran, Python
- Open – source

Nevýhody

- Nový jazyk, tedy i malá komunita a proto nabízí méně, než ostatní jazyky
- Velmi malá podpora pro tvorbu grafického rozhraní

6.1.3 Python

Python [31] je dalším z řady skriptovacích jazyků. Od obou výše zmíněných se liší tím, že je všestranně zaměřený. Jeho syntaxe je navržena tak, aby se dala lehce naučit a aby byl kód velmi čitelný. Python je open-source a řadí se mezi nejoblíbenější programovací jazyky. Python vznikl z vize, kde jednoho dne bude znalost programovacího jazyka nutností. Díky jeho silné komunitě má Python knihovny se zaměřením na všechny odvětví. Se znalostí jazyka Python se rozšiřují možnosti.

Standardní Python je pomalý, avšak díky knihovnám tzv. „třetích stran", jako je například NumPy pro numerické výpočty, dosahuje v některých případech rychlostí jazyka C. Python též není plně objektově orientovaný jazyk. Důvodem je filozofie Guido Van Rossuma, který jej stvořil. Python neuznává jeden z hlavních bodů OOP a to zapouzdření. Narozdíl od Matlabu a Julie je však práce s třídami a objekty velmi jednoduchá a intuitivní a je podobná jako v Javě.

Výhody:

- Všestrannost
- Velmi jednoduchá syntaxe, čitelnost
- Objektově orientovaný
- Open – source
- Možnost tvorby multiplatformního grafického rozhraní

- Knihovny pro numerické a maticové výpočty
- Python na mikrokontrolérech (MikroPython)
- Python na jednodeskových počítačích (Raspberry pi apod.)

Nevýhody

- Problém s kompatibilitou různých verzí Pythonu (verze 2. a 3.)
- Časté komplikace při instalaci různých knihoven pro chod programu, který takovéto knihovny používá (Vyváženo např. Python distribucí Anaconda, která obsahuje nejrozšířenější knihovny a ulehčuje instalaci ostatních knihoven)
- Standarní Python je pomalý oproti např. jazyku C nebo Java

6.2 Porovnání SW nástrojů pro grafická rozhraní

Mé zkušenosti a dovednosti pokrývají tvorbu grafického rozhraní v Matlabu a Pythonu. Proto zde uvedu pouze tyto dvě možnosti.

6.2.1 Matlab

Matlab má velmi účinný nástroj pro tvorbu grafického rozhraní. Počítá s tím, že majoritní skupina koncových uživatelů tohoto softwaru jsou studenti, inženýři a vědecko – výzkumní pracovníci. Nikoliv však programátoři v pravém slova smyslu. Pro tvorbu grafického rozhraní tedy stačí přesouvat interaktivní prvky z palety na plátno grafického rozhraní. Pomoci Matlab jazyka se pak vytvoří logika a napojí se skript na grafické rozhraní. Tento nástroj je velmi intuitivní a lze sním vytvářet aplikace velmi rychle. Dalším kladem je možnost propojení např. modulu Simulink s logikou grafického rozhraní.

Slabinou je rigidita interaktivních prvků a malé možnosti toho nástroje. Obecně vzato lze říct, že účel grafického rozhraní Matlabu nejde za hranice

ovládání laboratorních úloh a tím pádem ani jeho možnosti nenabízejí víc, než je potřeba k tomuto účelu.

6.2.2 Kivy

Kivy je Python knihovna pro tvorbu mutliplatformních grafických rozhraní s moderním vzhledem (Windows, Linux, Mac, iOS, Android). Vývoj Kivy je orientován především na požadavky budoucnosti. Kivy podporuje více-dotykové ovládání a je open-source. Je pod licencí MIT, která dovoluje využít Kivy ke komerčním účelům. Design interaktivních prvků grafického rozhraní lze vytvořit dle libosti. V Kivy lze tvořit jak hry, tak i inženýrské aplikace. Kivy tak nabízí lidem se základními programátorskými dovednostmi možnost ovládat své okolí ze smartphonů a tabletů [32].

Kivy jazyk je jazyk, kterým se tvoří uspořádání grafického rozhraní a propojují se funkce v Python skriptu s interaktivními prvky Kivy. Pomocí tohoto jazyka lze navrhnout vzhled a strukturu komplexního grafického rozhraní za 10 - 15 minut.

6.2.3 Volba vývojového prostředí

Vývojová prostředí, která nejvíce vyhovují naprogramování ladícího nástroje a jsou zároveň vhodnými pro náročné výpočty, jsou Matlab a Python. V obou případech lze relativně rychle tvořit a možnosti obou prostředí jsou pro daný účel široké.

Pro zpracování zadání diplomové práce jsem však zvolil Python. V několika bodech zde odůvodním svoji volbu.

- Neuronové sítě jsem naprogramoval objektově (nad rámec zadaní), tedy tak, aby práce s nimi byla jednodušší a víceméně automatizovaná. Z pohledu objektově orientovaného programování je Python nadřazenější.

- Python se pro mne nedávno stal hlavním programovacím jazykem a využívám ho i pro své vlastní projekty. Objektově naprogramované neuronové sítě tedy mohu použít i v budoucnosti.
- Python je open-source. Všechno co jsem v diplomové práci stvořil, budu moct nadále používat a rozšiřovat.
- Možnosti grafického rozhraní jsou nesrovnatelné. Znalost Kivy otevírá velké možnosti při realizaci vlastních projektů a každá nově nabitá dovednost v Kivy člověka v tomto směru posouvá dál.

7 Energo Data

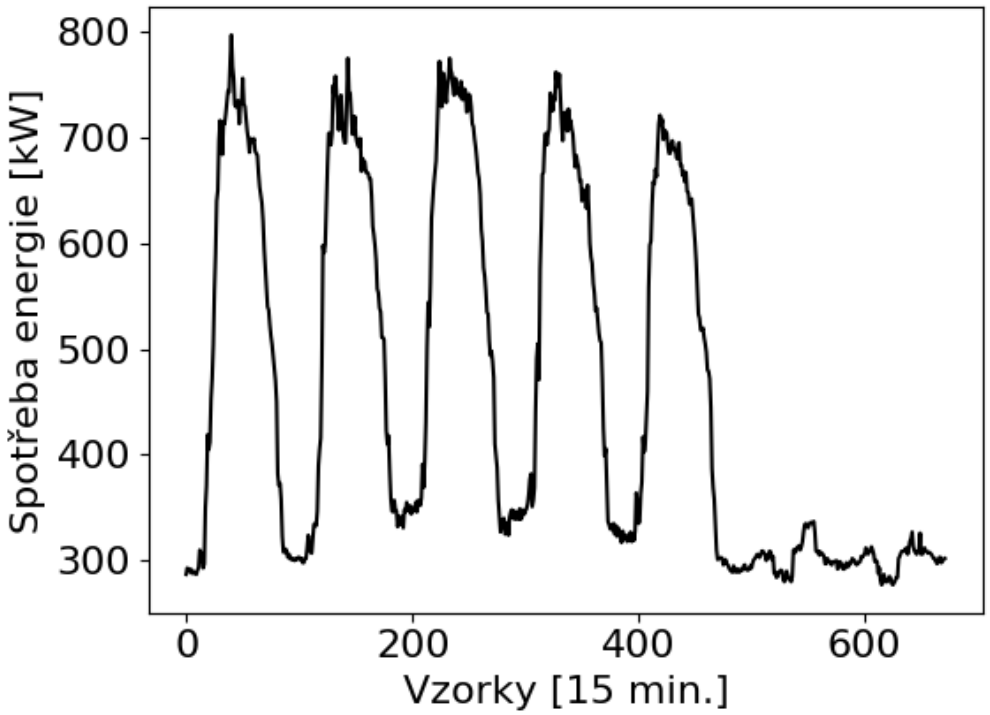
Od společnosti ENERGOCENTRUM Plus jsem získal data, která vyjadřují spotřebu energie (kW) celkem 11 budov v časovém intervalu přibližně jednoho roku. O Charakteru spotřeby toho není moc známo. Součástí dat je pouze spotřeba energie a korespondující datum a čas. Vzorkování dat má periodu 15 minut. Rozlišovací schopnost dat je 1kW. V tomto ohledu budovy s větší spotřebou mají kvalitnější data a obsahují v sobě citlivější informace o dynamice spotřeby. V této kapitole seznámím čtenáře o významu a charakteru dat z mého pohledu a dle mého názoru, z kterého bude dále vyplývat metoda predikce spotřeby energie budov a detekce neobvyklých stavů.

7.1 Úvaha o významu dat a možnostech teoretického modelu

7.1.1 Budova jako organismus

Data spotřeby energie budov spolu s korespondujícím datem a časem samy o sobě nemusí nestačí k predikci spotřeby energie nebo k vystihnutí její dynamiky. Data od ENERGOCENTRUM Plus v sobě obsahují mnoho

závislostí různých řádů (vyplývá z velmi odlišné spotřeby energie jednotlivých budov).

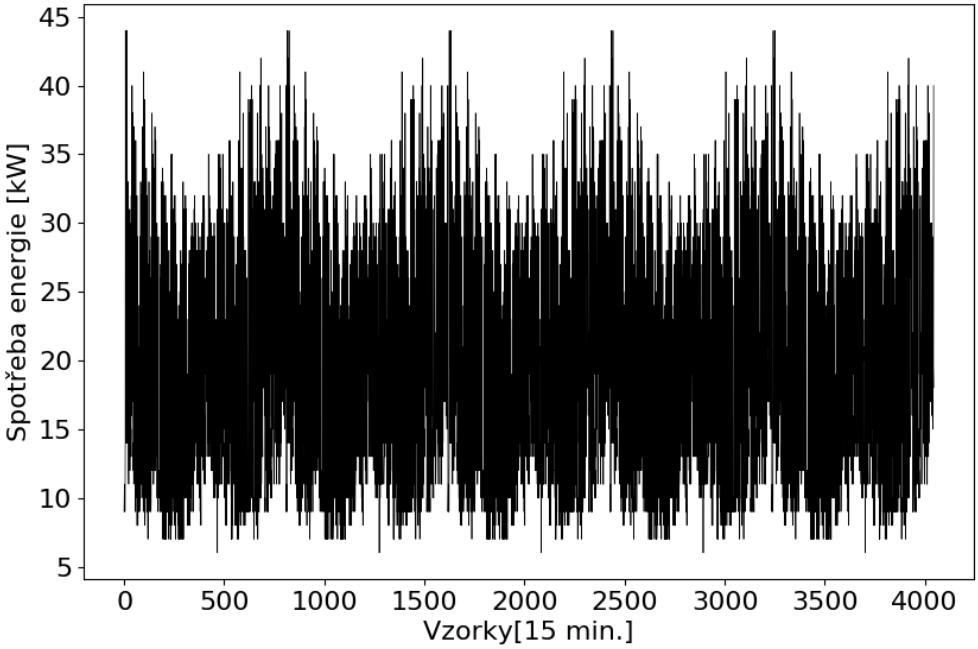


Obr. 7: Spotřeba energie budovy v intervalu jednoho týdne

Na Obr. 7 je vidět spotřeba konkrétní budovy v intervalu jednoho týdne. Vrcholky jsou pracovní dny, „údolí" je v intervalu mezi pozdním večerem a ránem a o víkendu je spotřeba energie v podstatě konstantní. Tato konkrétní budova má spotřebu energie i v pasivním stavu (v noci i o víkendu se drží spotřeba na 300kW). Cyklus týdne této konkrétní budovy a budov ostatních se opakuje celý rok a jejich charakter cyklu týdne je v kratším časovém intervalu téměř neměnný.

Predikovat lze jen tam, kde je systém. Bez závislostí nejde předurčit nic a pokud ano, je to jen náhoda. To samé platí v případě detekce neobvyklých stavů – nevím-li co je chyba, nelze detekovat ani to co chyba není. Zde uvedená data bezpochyby vykazují známky systému a proto se pokusím pomocí úvahy najít závislosti tohoto systému, které by mohly posloužit jako vstup do neuronové sítě.

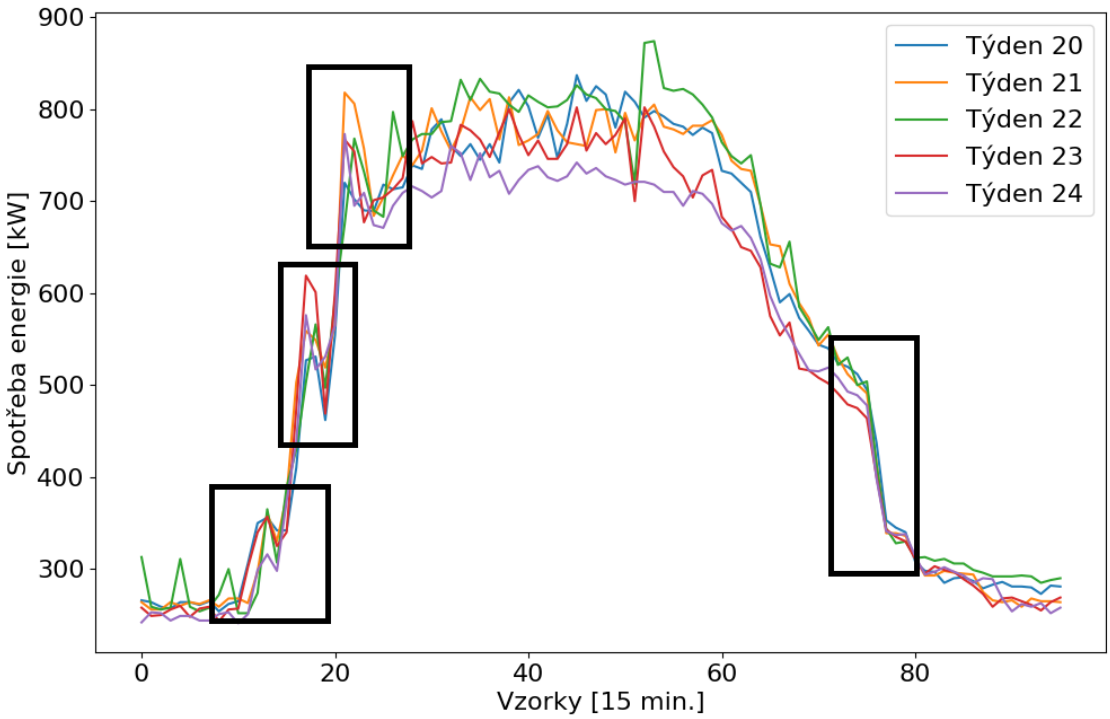
Budova jako taková slouží zájmům člověka a proto dynamika spotřeby energie budovy je v přímé závislosti s denní činností lidí. Činnost lidí je zas v přímé závislosti na okolním prostředí, tedy přírodě. Příroda pokrývá Zemi a za vše co se děje v přírodě je odpovědné Slunce (poloha Země vzhledem k Slunci určuje úhel, pod kterým dopadá sluneční záření, tedy energie na Zem) a dále rotace Země okolo vlastní osy. Toto je takříkajíc pohled shora. Budova je v tomto ohledu jako organismus, který reaguje na vnější podněty a zároveň je závislý na svých vlastních *vnitřních funkcí*. Prodloužil jsem data jedné z budov (Obr. 8), tak



Obr. 8: Uměle vytvoření pěti let spotřeby energie

abych získal interval pěti let. Na obrázku je vidět poloha Země vzhledem ke slunci (obálka dat, jejíž průběh se stále opakuje) a přiblížením dat bych získal předchozí obrázek (Obr. 7), kde je vidět vliv rotace Země kolem své vlastní osy (den a noc). Je tedy zřejmé že budova jako organismus z pohledu „shora" tepe v několika rytmech – roku, týdne a dne. Roční rytmus mění teplotu, která je zásadní veličina z hlediska spotřeby energie a z Obr. 8 ji lze spatřit jako dolní nebo horní obálku dat (v dolní obálce lze zahlédnout 4

roční období). Týdenní a denní rytmus vychází naopak z lidské činnosti jako takové a spotřeba energie v tomhle ohledu nebude závislá pouze na teplotě (vytápění či chlazení) ale i na vnitřním uspořádání (resp. řízení) budovy, čili na *vnitřních funkcí* organismu na které budu nahlížet z pohledu ‚zdola'. O těchto *vnitřních funkcí* nemůžu nic vědět a i kdybych je znal, tak je jen velmi těžko dám do společné závislosti. Patří mezi ně spotřeba energie spotřebičů a zařízení, které mohou být nebo nemusí být obsluhovány člověkem. Vnitřní řad budovy, tedy pracovní doba. Firemní zakázky či jakékoliv události většího rozsahu, které okamžitě zapůsobí na člověka, což se projeví například v narušení pracovní doby a tedy i denního rytmu budovy. Dále pak vnitřní řád a pravidla, která panují v budově. Naopak třeba spotřeba energie v období svátku by měla mít za normálních podmínek stejný charakter jako o víkendech, neboť z hlediska lidské činnosti jsou si tyto dva dny rovny.



Obr. 9: Spotřeba energie jednoho dne v intervalu 5 týdnů

Na Obr. 9 lze vidět pohled „zdola", tedy *vnitřní funkce* organismu budovy. V grafu je zobrazeno pondělí v pěti týdnech. Okamžitě lze zahlédnout několik *vnitřních funkcí* konkrétní budovy.
První tři *funkce* se projevují jako překmit v časovém intervalu mezi 3. až 9. hodinou (vzorky 10 až 30). Další funkce je v čase okolo 20. hodiny (vzorek 75) a zvláštní na ní je, že spotřeba energie *dokonale* splývá pro všech 5 týdnů.

Znalost *vnitřních funkcí* je velmi výhodná, neboť její příčinu lze do určité míry vyjádřit jako vstupní veličinu do neuronové sítě a tím pádem je možné získat model, který zná více stavů. Při neznalosti *vnitřních funkcí*, jako v mém případě, nelze jinak než předpokládat, že denní chod budovy je každý den stejný.

7.2 Veličiny

S každou budovou, chceme-li zachytit její *vnitřní funkce*, je potřeba do určité meze zacházet individuálně. Je zřejmé, že s více veličinami, na kterých je spotřeba energie přímo závislá pro konkrétní budovu, bude predikce i detekce přesnější. Data od ENERGOCENTRUM Plus neobsahují kromě spotřeby energie žádná jiná konkrétní data k jednotlivým budovám. Proto vzhledem k předešlé úvaze jsem navrhl veličiny, na kterých jsou budovy závislé z pohledu „zhora" i „zdola" a které by měli být schopné vystihnout (obecnou) dynamiku spotřeby energie jakékoliv budovy.

7.2.1 Čas

Čas je tou nejsilnější veličinou, která je k dispozici. Čím je budova větší a organizovanější, tím lépe lze zachytit řád a pravidla, které ovládají *vnitřní funkce* budovy.

Vstupem do neuronu budou dva druhy veličin, které vystihují čas různým způsobem. První veličina rozděluje den do úseků v intervalu (0,1). Další veličinou je rozlišení pracovních dnů a víkendu. Pro všední dny veličina nabývá hodnoty 1 a pro víkend nabývá hodnoty 0.

7.2.2 Spotřeba energie

Pro postihnutí dynamiky spotřeby energie využiji samotné naměřené hodnoty a to spotřebu energie korespondující s daným okamžikem včetně spotřeby energie, která předcházela.

7.2.3 Teplota

Teplota prostředí hraje všeobecně významnou roli při spotřebě energie a to jako příčina vytápění nebo chlazení. Teploty v České republice odpovídají teplotám mírného klimatického pásu a dle statistik průměrných teplot v rozsahu celého roku [33] lze tuto závislost jednoduše a celkem přesně aproximovat funkcí sinus v intervalu $(0, \pi)$. Tato aproximace sice vystihuje charakter postupu teploty v ročním intervalu, avšak již nezahrnuje teplotní výkyvy. Bohužel se mi nepodařilo sehnat detailní meteorologické statistiky a proto v této práci použiji pouze aproximaci. V aplikaci jsem však aproximovanou teplotu nepoužil jako standardní vstupní veličinu, ale jako dodatečnou (Free mod) neboť se ukázalo, že na predikci konkrétních dat nemá příliš velký vliv.

8 Aplikace na predikci časových řad a detekci neobvyklých stavů

V této kapitole uvedu metody, které používám v aplikaci k predikci časových řad neuronovými sítěmi a k detekci neobvyklých stavů. V další části kapitoly popíšu aplikaci z hlediska jejích funkcí, možností a jejího ovládání.

8.1 Metoda predikce

Metodu predikce jsem zvolil tzv. Sliding window retraining[4] (Obr. 10). V této metodě se zvolí šířka okna (win) a horizont predikce (p). Mám vstupní matici $\mathbf{X^M}$, jejíž řádky reprezentují vstupní vektor do neuronu a dále mám vektor reálných (naměřených) dat y_t. Rozměr vstupní matice je $L \times n$, kde n je velikost vstupního vektoru do neuronu včetně biasu (vychýlení) a rozměr vektoru reálných dat je L.

Vstupní vektor do neuronu , který je součástí vstupní matice, obsahuje reálná data a jejich nedávnou minulost dle (8.1). Obsahuje též další vstupní veličiny (viz 7.2.1), ale pro názornost je zde vynechám.

$$\mathbf{X^M} = \begin{bmatrix} 1 & y_t(k) & y_t(k-1) & y_t(k-2) & \cdots & y_t(k-n+1) \\ 1 & y_t(k-1) & y_t(k-2) & y_t(k-3) & \cdots & y_t(k-n+2) \\ 1 & y_t(k-2) & y_t(k-3) & y_t(k-3) & \cdots & y_t(k-n+3) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & y_t(k-win) & y_t(k-win-1) & y_t(k-win-2) & \cdots & y_t(k-win-n+1) \end{bmatrix} \quad (8.1)$$

Výstupem z neuronu je vektor vypočítaných hodnot (v případě dávkového učení)

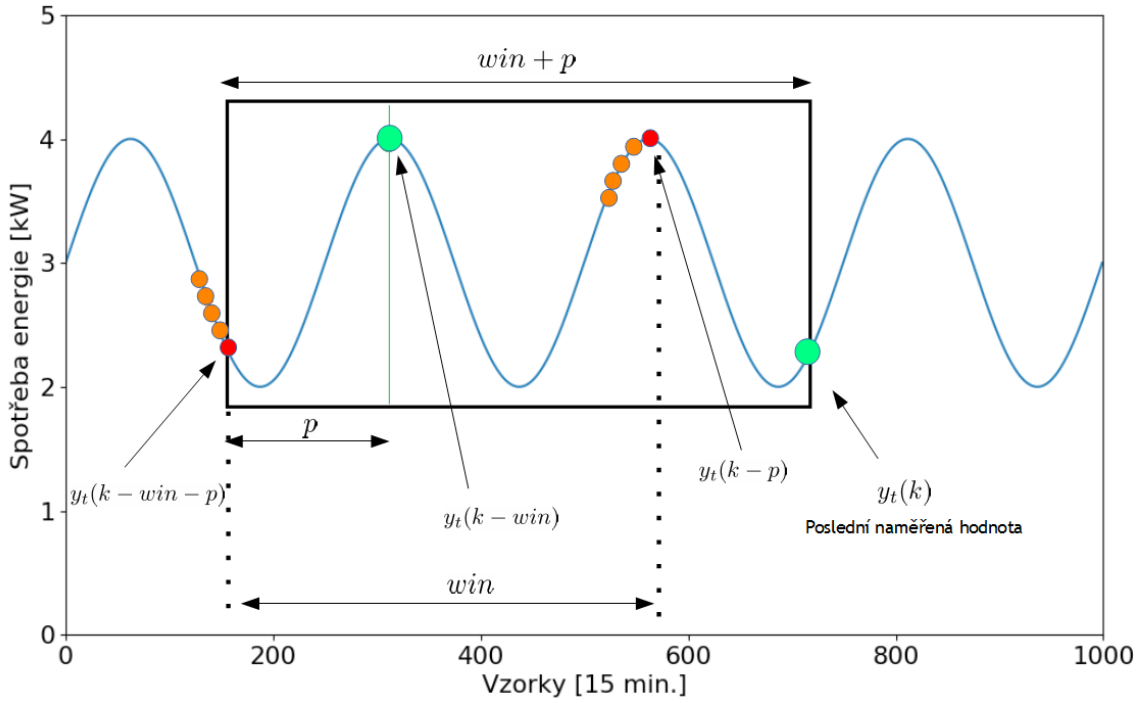
$$\mathbf{y_n} = \begin{bmatrix} y_n(k+p) & y_n(k+p-1) & y_n(k+p-1) & \cdots & y_n(k+p-win) \end{bmatrix} \quad (8.2)$$

V případě krokového učení je výstupem z neuronu číslo

$$y_n(k+p) = f(\mathbf{x^V}(k), \mathbf{W}), \quad (8.3)$$

kde $\mathbf{x^V}(k)$ bude v tomto případě vstupní vektor.

4 Sliding window retraining - metoda klouzájícího okna



Obr. 10: Princip predikce metodou klouzajícího okna, tak jak jsem jí použil v aplikaci. Zvolí se šířka okna (win) a horizont predikce (p). Červený kruh představuje konkretní hodnotu, z které vyplývá vstupní vektor obsahující zpožděné naměřené hodnoty – oranžové kruhy. Zelený kruh představuje naměřenou hodnotu, na kterou se vstupní vektor učí.

Z hlediska krokového učení je potřeba každý vstupní vektor naučit na reálnou hodnotu v celém rozsahu okna. Bude-li $y_t(L)$ poslední naměřená hodnota, pak první vstupní vektor klouzajícího okna $\mathbf{X^V}(L - win - p)$ se učí na $y_t(L - win)$ a poslední $\mathbf{X^V}(L - p)$ se učí právě na poslední naměřené hodnotě $y_t(L)$. Z hlediska dávkového učení je pojem klouzající okno zavádějící neboť se vstupní matice $\mathbf{X^M}$ učí přímo na vektoru reálných hodnot, tedy jinými slovy – učí se celé okno bez klouzání.

8.2 Metody detekce chyb

V této podkapitole popíšu postup a metody, kterými do určité míry detekuji chyby. Je-li neuronová síť natrénovaná na datech a již se neučí, pak je jediným úkolem detekovat chyby v datech. Vyžadují-li data přetrénování

neuronové sítě v určitých datových intervalech, pak je potřeba kromě detekce chyb zavést i detekci učení na chybných datech.

V případě dat od ENERGOCENTRUM Plus je nutné zavést i detekci učení na chybných datech, neboť dynamika spotřeby energie se mění s časem a proto je potřeba neuronovou síť pořád učit.

8.2.1 Metoda detekce učení na chybných datech

Název metody zní ***Learning Entropy for Novelty Detection.*** V této metodě dle [34] se detekuje velikost změny neurálních vah. Při každé změně narůstá entropie.

Přírůstek vah Δw získám z (5.5). Dále zvolím vektor citlivosti detekce

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix}, \tag{8.4}$$

kde $\alpha_1 > \alpha_2 > \cdots > \alpha_n$.

Zda je přírůstek vah neobvykle veliký se určí z podmínky

$$|\Delta w_i(k)| > \alpha_j \overline{|\Delta w_i(k)|}, \tag{8.5}$$

kde $\overline{|\Delta w_i(k)|}$ je průměrný přírůstek vah za určité období. Výpočet změny entropie se pak vypočte jako

$$E_A = \frac{1}{n_w \cdot n_\alpha} \sum_{j=1}^{n_\alpha N(\alpha_j)}, \tag{8.6}$$

kde n_w je rozměr vektoru vah, n_α je rozměr vektoru citlivosti detekce a $N(\alpha_j)$ je množství neobvykle velkých přírůstků vah. Změna entropie může nabývat pouze hodnot $E_A \in \langle 0, 1 \rangle$.

V mém programu tato metoda slouží jako doprovodný nástroj, díky kterému může uživatel určit, ve kterých intervalech je potřeba vypnout učení neuronové sítě. Funkci v Pythonu na spočítání entropie jsem převzal od doc. Ing. Bukovského Ph.D..

Poznámka k programu:

Kvůli přehlednosti a lepší čitelnosti se velikost změny entropie v grafu bude pro dané přetrénování držet po dobu, dokud nedojde k novému přetrénování.

8.2.2 Metoda detekce chyby v datech

Metodiku hledání chybných stavů jsem zvolil dle vlastního uvážení. Nejprve se zvolí počet hodnot n vypočítaných neuronovou sítí, ze kterých se určí chyba pro aktuální naměřenou hodnoty k. Dálé pak dle (8.7) získám výslednou průměrnou chybu.

$$e_{average}(k) = \frac{\sum_{i=0}^{n} |y_{neuron}(k-i)|}{n} \tag{8.7}$$

Chybný stav určím z poměru chyby $e_{average}(k)$ k uživatelem definované hodnotě D při splnění podmínky (8.8), kde q je navolená procentuální hodnota nedovolené chyby.

$$100 \cdot \frac{e_{average}(k)}{D} > q. \tag{8.8}$$

Tento způsob tvoření chyby má tu výhodu, že velikost chyby závisí na předešlých chybách a tím pádem se identifikují neobvyklé stavy trvající delší dobu. Volbou n tedy nastavujeme paměť chyby.

Součástí programu je též možnost vyhodnocení RMSE[5],MAE[6]
kde RMSE je odmocnina z průměru kvadrátu chyb predikce

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=0}^{N}(y_t - y_n)^2}, \tag{8.9}$$

kde MAE je průměrná absolutní chyba predikce

$$MAE = \frac{1}{N}\sum_{k=0}^{N}|y_t - y_n|. \tag{8.10}$$

5 RMSE – Root mean square error

6 MAE – Mean absolute error

8.3 Úprava dat

U některých dat od ENERGOCENTRUM Plus pomáhá k lepší predikci úprava dat vyhlazením pomocí metody Coarse Graining (CoGr)[7] dle

$$y_{cg}(k) = \frac{1}{2r+1} \sum_{i=-r}^{2r+1} y(k-i). \tag{8.11}$$

V této metodě se nastavuje, z kolika hodnot r (radius) okolo hodnoty $y(k)$ se vypočte aritmetický průměr, který se dosadí za hodnotu $y(k)$. Tato metoda se v mé práci osvědčila, neboť se po úpravě CoGr zlepšila predikce. Je však důležité nepřehánět s parametrem r. Doporučené hodnoty v této práci jsou pro $r \in (0, 10)$, kde $r = 0$ znamená zachování původních hodnot.

Dále je potřeba data normalizovat. Jeden z hlavních důvodů je, aby se srovnaly vlivy jednotlivých vstupních veličin tak, aby vstupní veličiny pohybující se v různých řádech jednotek měly stejné váhy. V této práci jsem použil velmi primitivní avšak účinný způsob. Vstupní veličiny, které vycházejí z naměřených dat jsem vydělil jejich nejvyšší hodnotou a získal tak vstupy v intervalu $(0, 1)$. Tento způsob jsem zvolil kvůli tomu, že v datech se nevyskytují žádné extrémní maximální hodnoty, které by tímto způsobem znehodnotily zbytek dat.

8.4 Grafické rozhranní

Aplikace je primárně zaměřená na predikci dat od ENERGOCENTRUM Plus, je však možné ji použít na jakákoliv data s vlastním nastavením neuronových sítí díky Free modu, který jsem do aplikace přidal. Free mod spočívá ve skriptovacím okénku, ve kterém uživatel v jazyce Python přiřadí parametrům neuronové sítě hodnoty, vytvoří proměnou se vstupním vektorem (maticí) do neuronu a proměnou obsahující reálná data. V této kapitole popíšu i způsob, jak ovládat Free mod.

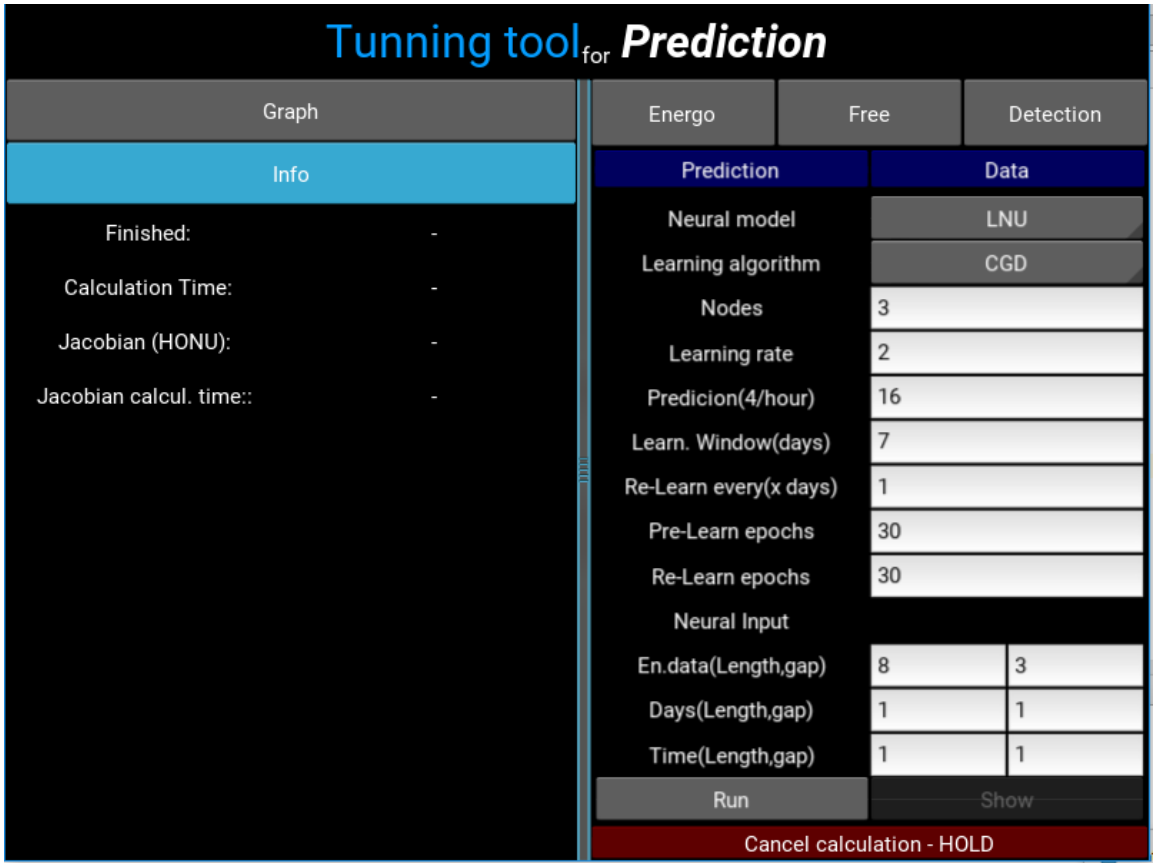
7 Coarse Graining – Metoda klouzavého průměru

8.4.1 Struktura

Grafické rozhraní (GUI) mé aplikace jsem navrhl tak, aby bylo přehledné a jednoduché na ovládání. Hlavní okno GUI obsahuje dvě podokna (Obr. 11).

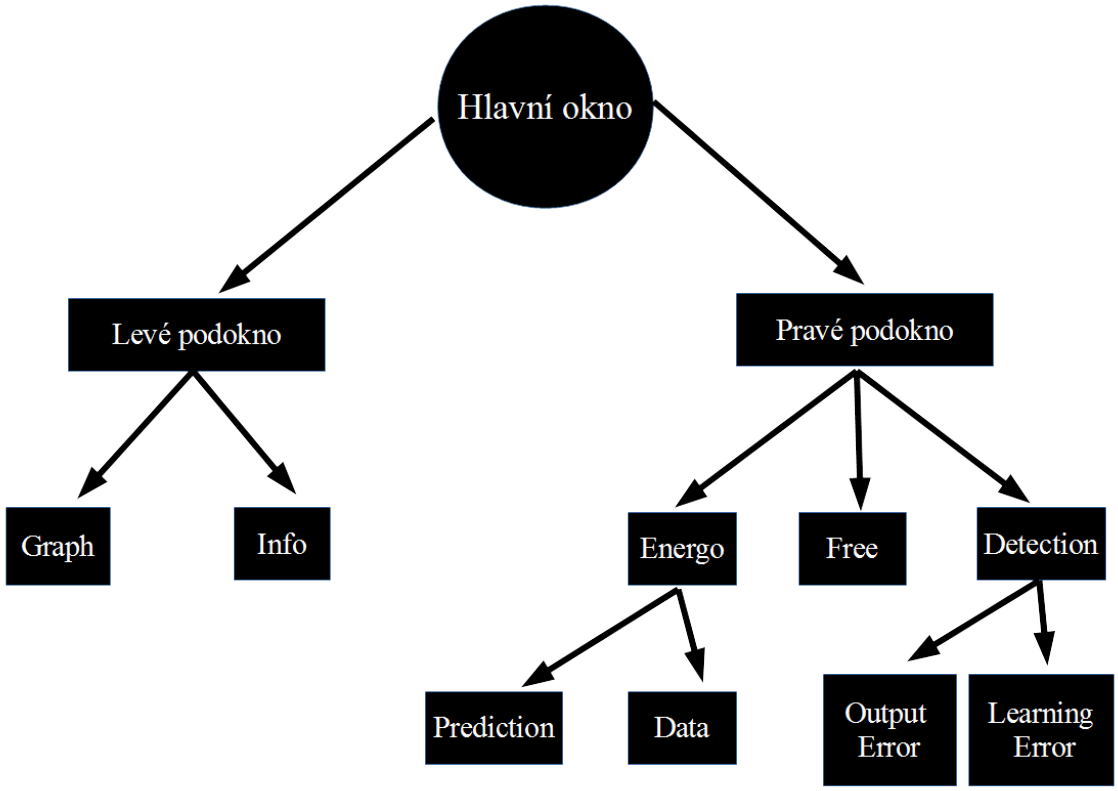
V prvním podokně nalevo lze nalézt 2 položky - **Graph**, tedy graf ve kterém se zobrazují data společně s výsledky neuronového výpočtu. Další položka je **Info,** ve které jsou údaje informující uživatele o výpočtu, který může být velmi rychlý, ale také velmi pomalý.

Druhé podokno je o něco složitější. Obsahuje menu s položkami **Energo, Free, Detection.** Každá položka obsahuje svoje vlastní podokna. V podokně **Energo** je další menu s dvěma položkami ***Prediction*** a ***Data.*** Podokno **Detection** má menu s dvěma položkami ***Output Error*** a ***Learning Error***.



Obr. 11: Vzhled GUI

Přehled struktury grafického rozhraní je vidět na Obr. 12.



Obr. 12: Struktura grafického rozhraní

8.4.2 Modul Energo – Predikce

V tomto okně se navolí tyto parametry:

- neuronová sít MLP nebo neuronová jednotka HONU – LNU, QNU, CNU
- učící algoritmus backpropagation
  - CG – Conjugate Gradient (pouze pro HONU)
  - LM – Levenberg-Marquardt
  - GD – Gradient Descent
  - NGD – Normalized Gradient Descent
- počet neuronů ve skryté vrstvě (pouze MLP)
- rychlost učení (LM, GD, NGD)
- horizont predikce
- velikost klouzajícího okna

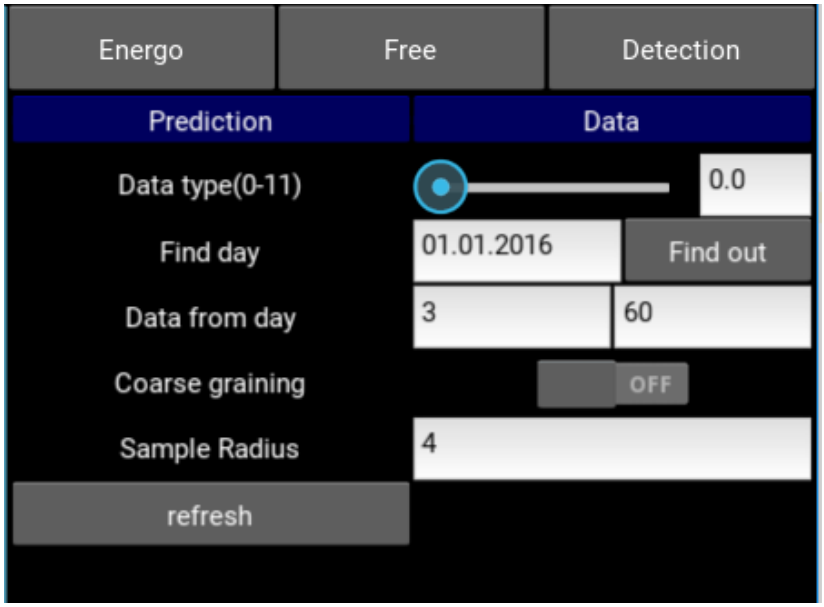
- interval přetrénovávání
- počet učících epoch pro předtrénování a přetrénovávání
- vstupní vektor
  - počet zpožděných naměřených hodnot a mezera mezi nimi, např. pro 3 zpožděné hodnoty a mezeru velikosti 3 bude mít vstupní vektor do neuronu tvar
    $\mathbf{x}(k) = \begin{bmatrix} y_t(k) & y_t(k-3) & y_t(k-6) \end{bmatrix}$
  - ke vstupnímu vektoru se připojí vstupní vektor dnů a času, jehož sestavení podléhá stejným pravidlům jako sestavení vstupního vektoru naměřených hodnot.

V dolní části se spouští výpočet (Run), zobrazuje se výsledek (Show results) a přerušuje se výpočet (Cancel calculation - HOLD).

8.4.3 Modul Energo – Data

V této podčásti se manipuluje s daty ENERGOCENTRUM Plus. Je zde několik funkcí které provádí tuto činnost:

- Posuvníkem se volí typ dat, tedy data konkrétní budovy(0-11, 11. jsou umělá data).
- Tlačítkem Find out lze nalézt konkrétní den.
- Výběr dat v intervalu mezi dvěma zvolenými dny
- Nastavení a povolení vyhlazení dat. Nastavuje se radius. Při vyhlazování se sečtou naměřené hodnoty v intervalu $\langle k - radius, k + radius) \rangle$ a za naměřenou hodnotu $y_t(k)$ se dosadí aritmetický průměr ze sečtených hodnot dle 8.11.
- Obnovení zobrazení dat



Obr. 13: Manipulace s daty v sekci Data

8.4.4 Modul Free mod

Zde je k dispozici celé skriptovací okno. Programuje se v Pythonu. V průběhu diplomové práce jsem vytvořil jednoduchou podporující knihovnu, jejíž používání je velmi jednoduché a díky ní lze používat **Free mod** i bez znalosti Pythonu. Učící algoritmus a typ neuronové sítě (neuronu) se zvolí v modulu **Energo - > Predikce,** ostatní nastavení musí být přiřazeno kódem.

Obr. 14: Modul Free mod

Uvedu zde stručně princip práce ve **Free modu** s daty ENERGOCENTRUM Plus. Mám 3 soubory ve formátu *txt* ve složce s programem, ve kterých jsou data ENERGOCENTRUM Plus a vstupní veličiny. Nahraju je příkazy

import *numpy as np #* numerická knihovna

import *neural_network as nn #* moje knihovna

yr = ***np.loadtxt****('ENE_data.txt') #* V textu značím **yr jako yt**

u1_dny = ***np.loadtxt****('log_days.txt')*

u2_cas = ***np.loadtxt****('time.txt')*

Je nutné, aby spolu data ze souborů korespondovala, tedy aby měly stejný rozměr. Mám tedy tři proměnné s třemi vektory. Naměřené hodnoty *yr* použiji i jako vstupní vektor.

X=yr

Všechny vstupní vektory nyní upravím podle potřeby a transformuju je tak, aby je kód přijal. Použiji k tomu funkce ze své knihovny

X = nn.force_recurrent_gap(X,5,3,0) # Naměřené hodnoty zpozdím o 5 hodnot s mezerou 3 a získám vstupní matici, se vstupními vektory (řádky matice). Jelikož vytvářím vstupní vektory z aktuální a minulých (zpožděných) naměřených hodnot, budou existovat vstupní vektory, které nemají minulé naměřené hodnoty (například první vstupní vektor). Poslední hodnota ve funkci tedy definuje hodnoty „neexistujících' naměřených hodnot.

U1 = nn.force_recurrent_gap(u1_dny,1,1,0) # naměřené hodnoty nezpozdím, ale musí projít touto funkcí.

U2 = nn.force_recurrent_gap(u2_cas,1,1,0) # naměřené hodnoty též nezpozdím.

Nakonec spojím vstupní matici X a vstupní matice (v tomto případě vektory) U1 a U2 do nové celkové vstupní matice X (obsahující matici se zpozděnými naměřenými hodnotami a vektory s korespondujícím časem a dnem)

X = nn.stack_input(X,U1)

X = nn.stack_input(X,U2)

Je důležité podotknout, že bias (vychýlení) se do vstupního vektoru přidává automaticky.

Má knihovna též nabízí vyhlazení dat (metodou klouzavého průměru) pomocí příkazu

yr =nn.coarse_grain(yr,3) # data yr jsou vyhlazeny s radiusem=3

Data, která jsou uložená pod proměnou yr musí být rozměrově 1D. Není-li v kódu chyba, pak je možnost zobrazit data yr nebo rovnou spustit výpočet.

Poznámky:

- *Tlačítky* **Save as** *a* **Load** *lze ukládat i nahrávat skripty ve formátu .txt. Jméno skriptu se pro oba účely zadává do textového pole vedle výše zmíněných tlačítek.*
- *.Tlačítkem* **save yn** *lze uložit vypočtené predikované hodnoty, které lze posléze v kódu zavolat příkazem: yn =* **np.loadtxt***("yn.txt")*
  Tato funkce slouží k tomu, aby bylo možné například nahradit neobvyklý stav predikovanou hodnotou.
- *Slabina Free modu je, že je-li chyba v kódu, není možnost zjistit její charakter. Proto doporučuji využívat šablony skritpů, které jsem přiložil do složky* **free mode code***.*

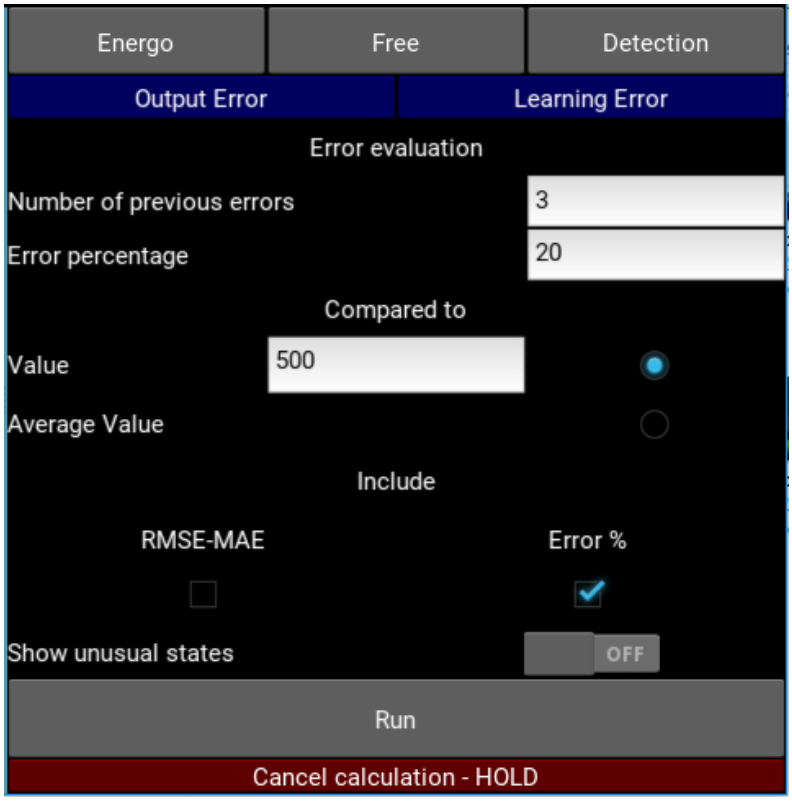
8.4.5 Modul Detection – Output Error

V tomto modulu (Obr. 15) se volí parametry, podle kterých se bude detekovat neobvyklý stav dat.
Mezi tyto parametry patří:

- Počet předchozích chyb, ze které se udělá výsledná průměrná chyba pro danou naměřenou hodnotu dle 8.11
- Velikost chyby vůči nadefinované hodnotě v procentech, která bude detekována
- Nadefinovaná hodnota zvolená uživatelem (doporučuji) nebo střední hodnota z dat (nedoporučuji, neboť například funkce sinus má tuto střední hodnotu 0, která způsobí, že rovnice (8.8) nemá řešení)
- Volba zobrazit RMSE-MAE a velikost chyby s detekčním limitem v grafu vyhodnocení
- Volba zobrazení detekovaných neobvyklých stavů

Pomocí tlačítka **Run se** zobrazí výsledky predikce s několika veličinami charakterizující chybu predikci s naměřenými daty(RMSE, MAE, a průměrnou

chybou dle (8.8)). Pro zobrazení detekčního signálu je potřeba povolit přepínač „Show unusual states"



Obr. 15: Modul s funkcí pro detekci chyb

Povolením vyhodnocení detekce chyb (**Show evaluation - ON**) se do výsledků predikce vyznačí, kde jsou neobvyklé stavy naměřených dat.

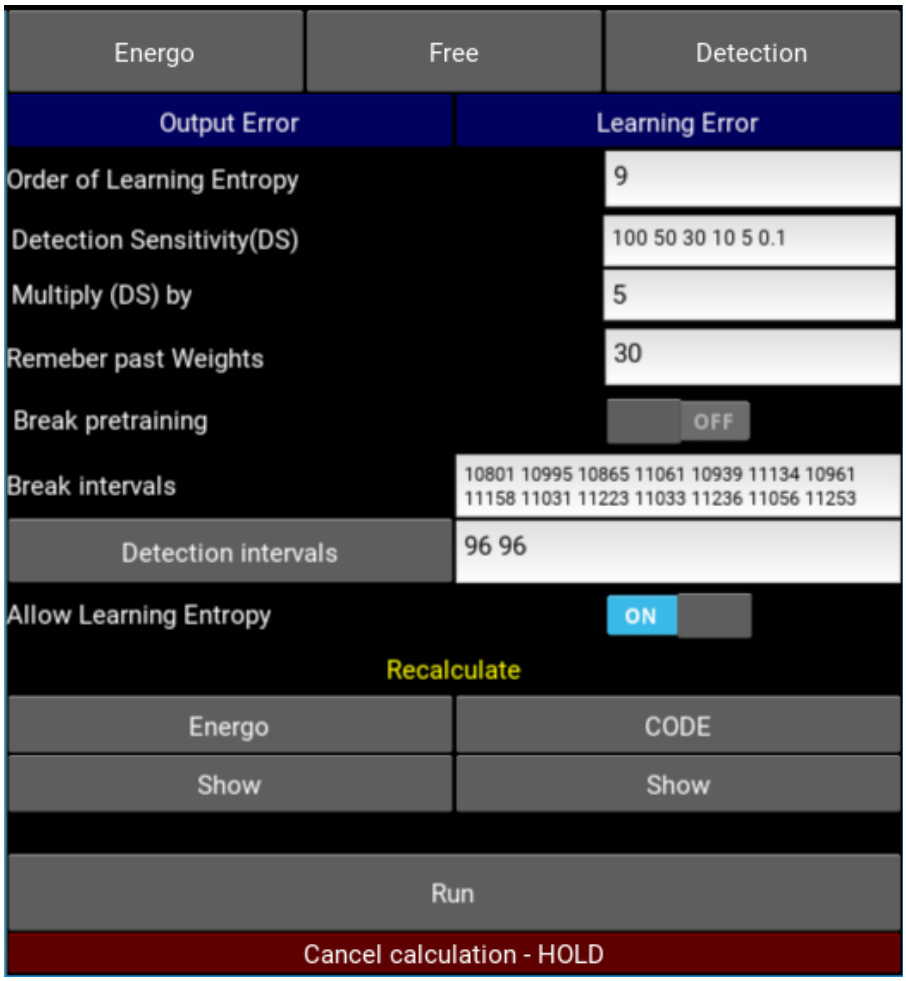
8.4.6 Modul Learning Error

Modul Learning Error (Obr. 16) využívá metodu ***Learning Entropy for Novelty Detection.*** Účelem této funkce je detekovat neobvyklé změny v učení neuronové sítě. Tyto změny mohou být jednak důsledkem neobvyklých stavů naměřených dat a nebo změnou dynamiky systému ze kterého plynou naměřená data. Tato funkce nabízí několik možností:

- Nastavit parametry metody *Learning Entropy for Novelty Detection*
  - Řád učení etropie (**Order of learning Entropy**)
  - Citlivost detekce (**Detection sensitivity, Multiply by**)
  - Pamět vah (**Remember past weights**)

- Přerušit učení v uživatelem definovaných intervalech (**Break pretraining - ON**)
- Tlačítkem Detection intervals se automaticky vygenerují intervaly pro přerusení na základě předchozího výpočtu a vyhodnocení .
- Povolit metodu v průběhu výpočtu, neboť značně zpomaluje výpočet u rychlých výpočtů (**Allow Learning Entropy**)
- Přepočet Energo dat i data z Free modu pro snadnější ladění parametrů (**Recalculate**)

Tlačítkem **Run** se otevřou výsledky.



Obr. 16: Modul s metodou pro detekci chybného učení

Poznámky k programu

Zadávání intervalu, ve kterém se má přerušit přetrénování funguje tak, že se zadává libovolný počet dvojic čísel, kde každá dvojice odpovídá intervalu „od do". Hodnoty intervalu odpovídají vzorkům.

Příklad: 10 20 30 40 → intervaly přerušení jsou (10,20),(30,40)

Je nutné zadávat celá čísla. Tyto čísla odpovídají vzorkům v grafu.

Upozornění: Program automaticky přičte k intervalu také velikost trénovacího okna a horizont predikce. Je-li chyba v datech, kterou nechci neuronovou síť naučit, v intervalu (100,110), zadám přesně tyto hodnoty a program automaticky rozšíří interval (100,110+šířka trénovacího okna+ horizont predikce). Pokud uživatel používá naměřené hodnoty jako vstup do neuronu, pak musí k intervalu ještě připočítat pozici poslední zpožděné hodnoty. Je-li například ve vstupním intervalu naměřená hodnota yr(k-10), pak musí interval rozšířit o 10, tedy na interval (100,120).

Pro automatické vygenerování intervalu je nutné projít fází detekcí neobvyklých stavů (‚Show unusual states' - ON → Run), neboť vygenerované intervaly přesně odpovídají intervalům detekovaných neobvyklých stavů. V poli vedle tlačítka Detection intervals se zadává tolerance pro začátek a konec intervalu. Pro názornost předvedu:

Detekované intervaly: 100 200 530 590

Tolerance : 5 1

Vygenerované intervaly: 95 201 525 591

8.4.7 Stabilita grafického rozhraní

Program je velmi stabilní, je-li uživatel opatrný a dodržuje tyto pravidla:

- zadává hodnoty do textových polích ve správném formátu. Nabývá-li nějaký parametr pouze celočíselných hodnot, může program spadnout, bude-li zadána neceločíselná hodnota (ale 10 ==10.0)
- dodržuje kauzalitu. Snažil jsem se program obránit proti požadavkům, které nemůže splnit, avšak je velmi možné že program obsahuje celou řadu bugů. V tomto ohledu by například uživatel neměl žádat program o výsledky z výpočtu, který se ještě neprovedl.

- nekliká vícekrát na tlačítko výpočtu. Program je navržen tak, aby fungoval plynule a proto výpočty běží v paralelním vlákně. Nicméně jsem přidal tlačítko Cancel calculation – HOLD, které když se podrží, tak stornuje všechny probíhající výpočty.

Bohužel jsem našel dva bugy[8], které nedokážu odstranit. Jedná se o bug v grafu. Graf má interaktivní lištu s nástroji, se kterými lze manipulovat s grafem. Zároveň tato lišta vzniká a zaniká podle toho, jakou šířku má okno grafu (Aplikace má možnost měnit poměr dvou hlavních oken). Aplikace spadne, nechá-li uživatel aktivní „Lupu" a poté změní poměr oken tak, že lišta zmizí. Po znovu objevení lišty aplikace spadne, použije-li uživatel znova lupu. Řešení na bezpečné používaní jsou dvě. Vždy odkliknout Lupu po používání, nebo otevřít okno do modu fullscreen, ve kterém nebude mít lišta možnost zaniknout. Další bug vzniká při prohlížení dat. Okno grafu si po prvním výpočtu uloží meze os x a y, a od té doby je nezmění. Proto pro návrat do výchozího stavu grafu, kde jsou meze osy x a y dány daty je potřeba kliknout na tlačítko Show/Show results místo na „domeček", který je v liště grafu.

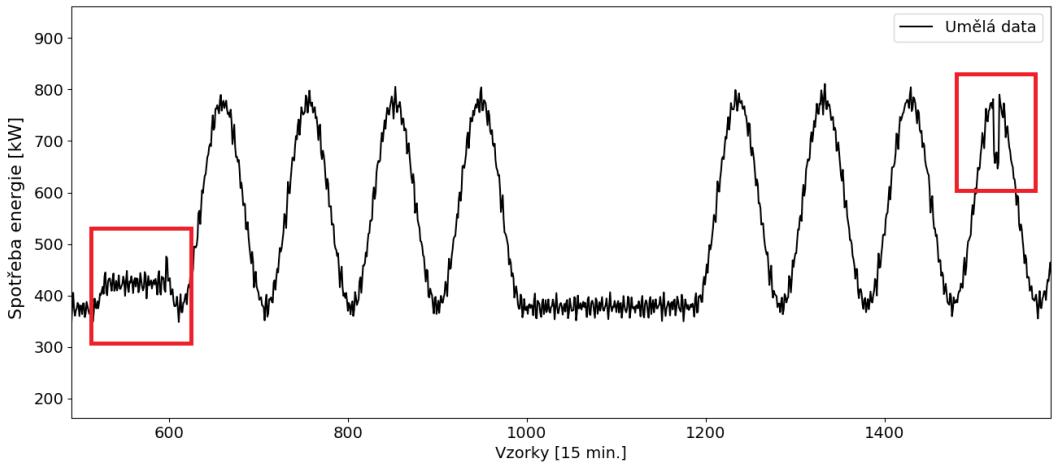
9 Ověření funkčnosti programu na umělých datech

V této kapitole ověřím funkčnost programu a jeho možností na umělých.

9.1 Umělá data

Vytvořil jsem umělá data, která napodobují data od ENERGOCENTRUM Plus. Kostra umělých dat tvoří kvadrát sinusové funkce. Na kostru je obaleno

8 Bug – chyba v programu



Obr. 17: Umělá data s označenými chybami

několik dalších harmonických signálů různých frekvenci a nakonec šum. Do umělých dat jsem přidal dvě chyby a pomocí programu tyto chyby detekuji a v dalším kole ošetřím neuronovou síť tak, aby se neučila v intervalech, kde se tyto chyby vyskytují. Na (Obr. 17) jsou vidět umělá data i s chybami. V Aplikaci jsem použil MLP síť s učícím algoritmem Levenberg-Marquardt. Nastavení parametrů dle Tab 2.

Neuronová síť	MLP
Učící algoritmus	LM
Neurony	3
Predikce	+16 vzorků
Rychlost učení	2
Trénovací okno	7 (672 vzorků)
Přetrénovávání	1(96)
Epoch naučení	30
Epoch přeučování	30

Tab 2: Parametry neuronové sítě, predikce a učení

Vstupní vektor jsem nastavil dle Tab 3.

	Počet zpožděných hodnot	Mezera mezi hodnotami
Naměřené hodnoty	8	3
Dny	5	96
Čas	1	1

Tab 3: Vstupní vektor

Detekci chyb dle Tab 4.

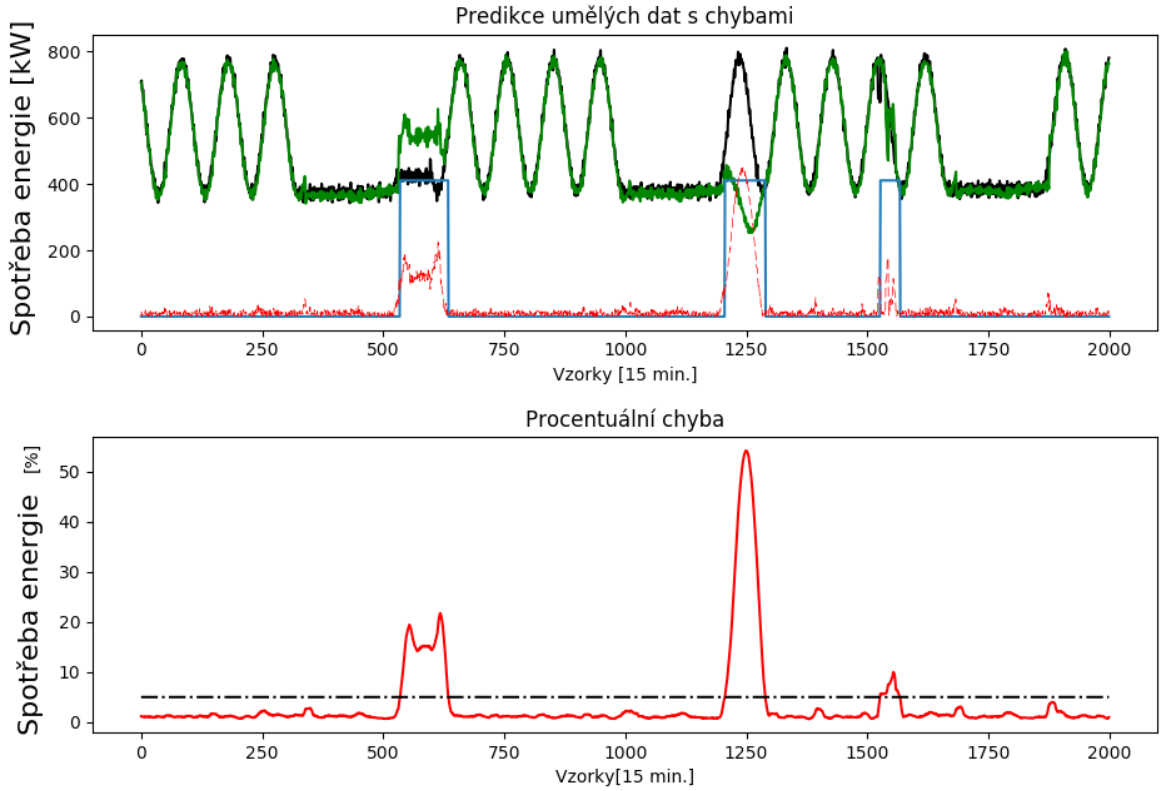
Počet vzorků ze kterých se vytvoří průměrná chyba	Hodnota, vůči které se bude srovnávat průměrná chyba	Limit detekce chyby
16	800kW	6%

Tab 4: Parametry detekce chyb

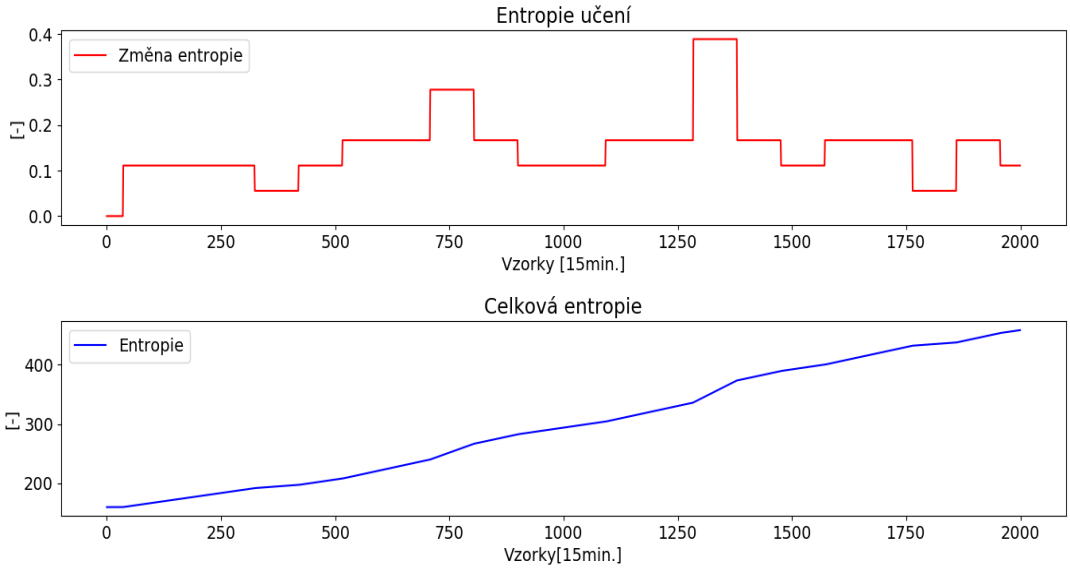
Parametry entropie učení dle Tab 5.

Řád entropie učení	Citlivost detekce	Paměť vah
4	80,40,24,5,4,0.08	15

Tab 5: Parametry metody Entropie učení



Obr. 18: Predikce a detekce neobvyklých stavů na umělých datech

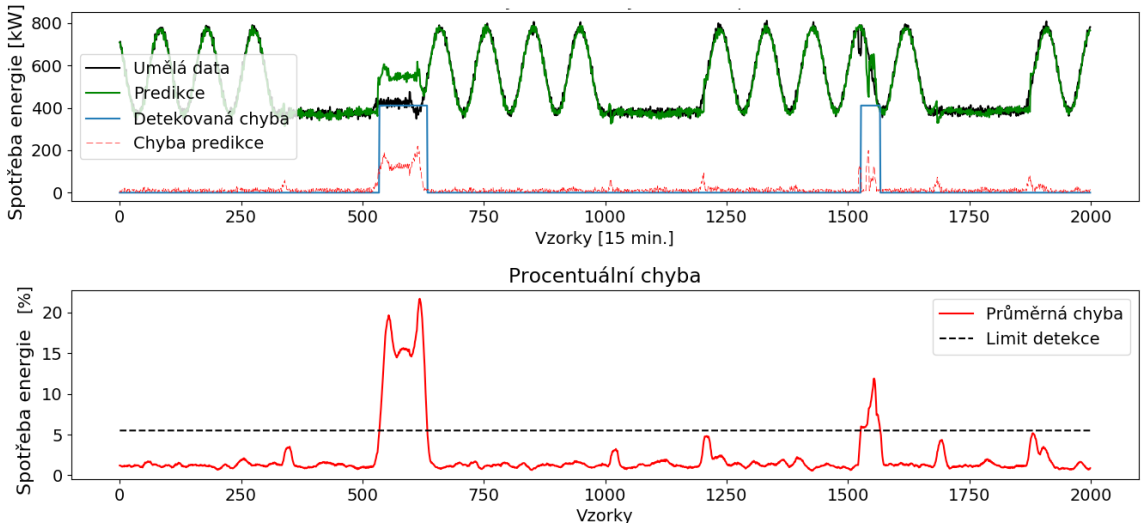


Obr. 19: Změna entropie učení a celková entropie učení

Výsledky predikce a detekce na Obr. 18 naznačují místo dvou chyb tři. To je způsobeno přetrénováním na chybných datech. Kromě chyby vzniklé chybným učením jsou zde další chyby, které však splývají se záměrně

vloženými chybami. Jsou způsobené tím, že predikuji z chybných naměřených hodnot. To způsobí chybnou predikci a tedy i detekci chyby. Zajímavý je výsledek detekce chybného učení na datech pomocí metody entropie učení (Obr. 19), který podle očekávání měl výrazně zvýšit entropii pouze jednou, nikoliv dvakrát, a to v okolí vzorků 750, tedy u první menší změny. To je dle mého vysvětlení způsobeno tím, že neuronová síť se učila na chybě týden a přesně po 7 dnech přišel standardní týden bez chyb a tudíž se neuronová síť učila znovu a zvýšila se entropie učení.

V aplikaci jsem nastavil přerušení učení v intervalu $(450, 650)$. Program automaticky nastaví, aby se neuronová síť neučila celý týden (je potřeba zajistit, aby od chyby v k se neuronová síť neučila ještě do vzorku $k + p + win + c$, kde c reprezentuje počet zpětných hodnot z reálných data, které jsou součástí vstupního vektoru. Program automaticky připočte jen $k + p + win$). Ve výsledcích na Obr. 20 jsou detekovány už jen dvě chyby, neboť se neuronová síť neučila na chybných datech a nezapříčinila novou chybu. Druhá chyba též způsobí chybné učení, proto lze na ní aplikovat stejnou metodiku jako na první chybu.



Obr. 20: Predikce a detekce neobvyklých stavů na umělých datech s vyřazenými intervaly učení

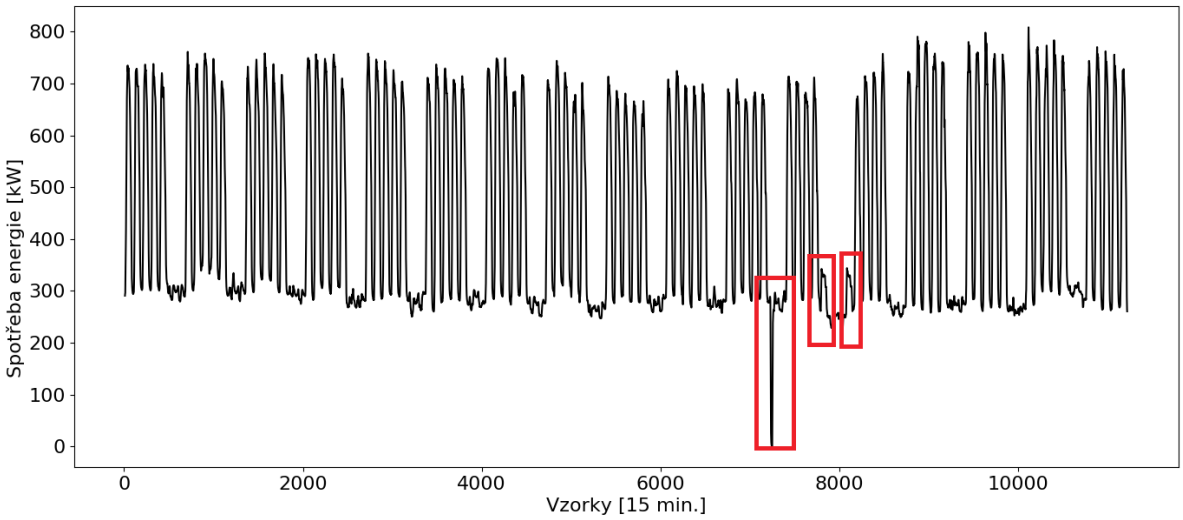
10 Dosažené výsledky na reálných datech

10.1 Problém reálných data

K dispozici jsou data z 11 různých budov. Většina z nich však má velmi proměnlivou a rozdílnou dynamiku spotřeby energie. Naměřené hodnoty mají u některých budov každý den jiný charakter. Vzhledem k počtu vstupních veličin do neuronu, které mám k dispozici (čas, dny, naměřené hodnoty), není možné predikovat příliš daleko. V dalších podkapitolách předvedu dosažené výsledky na budovách s rozdílnou spotřebou energie.

10.2 Budova s velmi stabilním průběhem spotřeby energie

První budovu, u které budu detekovat neobvyklé stavy je z hlediska dat ENERGOCENTRUM Plus 10. (počítáno 0-10, dohromady 11 bez umělých dat). Na Obr. 21 je zobren 3. – 120. den s označenými neobvyklými stavy.



Obr. 21: Spotřeba energie 117 dní budovy 10 s třemi vyznačenými neobvyklými stavy. -První stav je nejspíše výpadek energetické činnosti.- Dva další stavy vykazují sníženou energetickou činnost . Jedná se o období státních svátků (Velikonoce)

Jelikož průběh spotřeby energie je velmi stabilní, dovolím si predikovat spotřebu energie až 16 vzorků dopředu (4 hodiny, tedy p=16). Nejlepší predikční výsledky jsem získal z neuronové sítě MLP (nastavení dle Tab 6, Tab 7, Tab 8, Tab 9) s učícím algoritmem LM a o něco málo horší výsledky s kubickým neuronem HONU s LM a CG. Výpočet HONU je však 10krát rychlejší. Poté co mi přišly první výsledky, jsem označil místa, kde se neuronová síť učila na nesprávných datech a zadal interval, kde se síť nemá přetrénovávat a opět jsem přepočetl predikci i s detekcí chyb.

Nastavení parametrů dle Tab 6.

Neuronová siť	MLP
Učící algoritmus	LM
Neurony	3
Predikce	+16 vzorků
Rychlost učení	2
Trénovací okno	7 (672 vzorků)
Přetrénovávání	1 (96 vzorků)
Epoch naučení	30
Epoch přeučování	30

Tab 6: Parametry neuronové sítě, predikce a učení

Vstupní vektor dle Tab 7.

	Počet zpožděných hodnot	Mezera mezi hodnotami
Naměřené hodnoty	8	3
Dny	5	96
Čas	1	1

Tab 7: Vstupní vektor

Detekci chyb dle Tab 8.

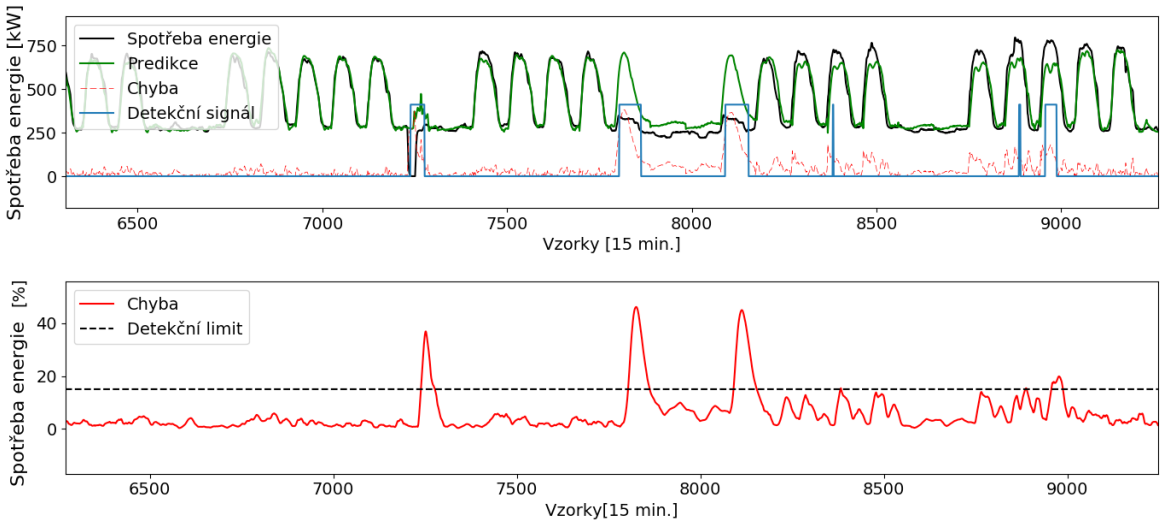
Počet vzorků ze kterých se vytvoří průměrná chyba	Hodnota, vůči které se bude srovnávat průměrná chyba	Limit detekce chyby
10	800kW	15%

Tab 8: Parametry detekce chyb

Parametry entropie učení dle Tab 9.

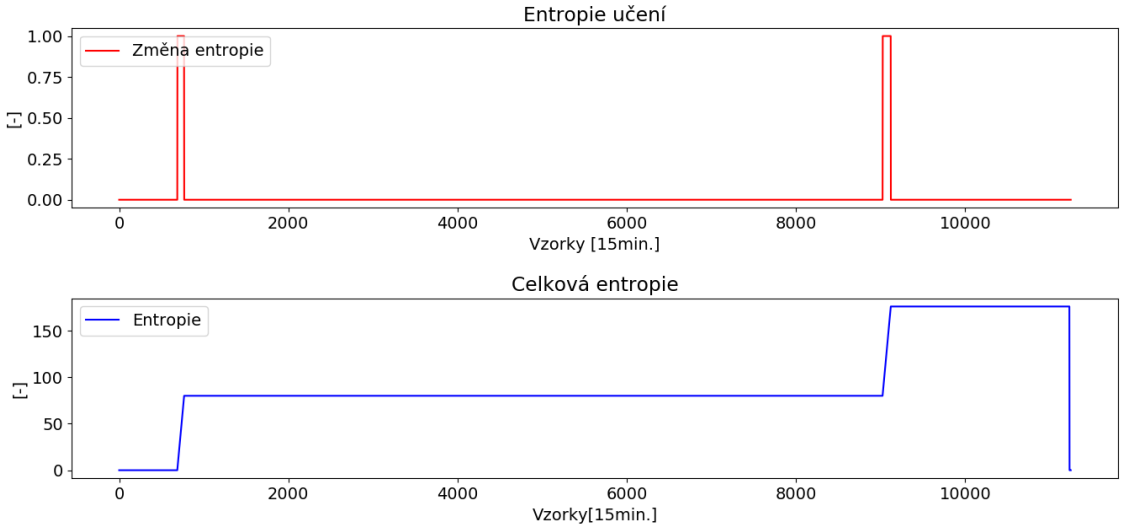
Řád entropie učení	Citlivost detekce	Paměť vah
4	80,40,24,5,4,0.08	15

Tab 9: Parametry entropie učení



Obr. 22: Detekce neobvyklých stavů. Z horního grafu je zřejmé, že všechny označené neobvyklé stavy(Obr. 21) byly detekovány. V dolním grafu je vidět procentuální chyba detekovaných neobvyklých stavů

Výsledky na Obr. 22 naznačují celkem 6 neobvyklých stavů (při 15% chybě predikce). První tři chyby odpovídají těm vyznačeným na Obr. 22 a jejich procentuální chyba je dominantní.



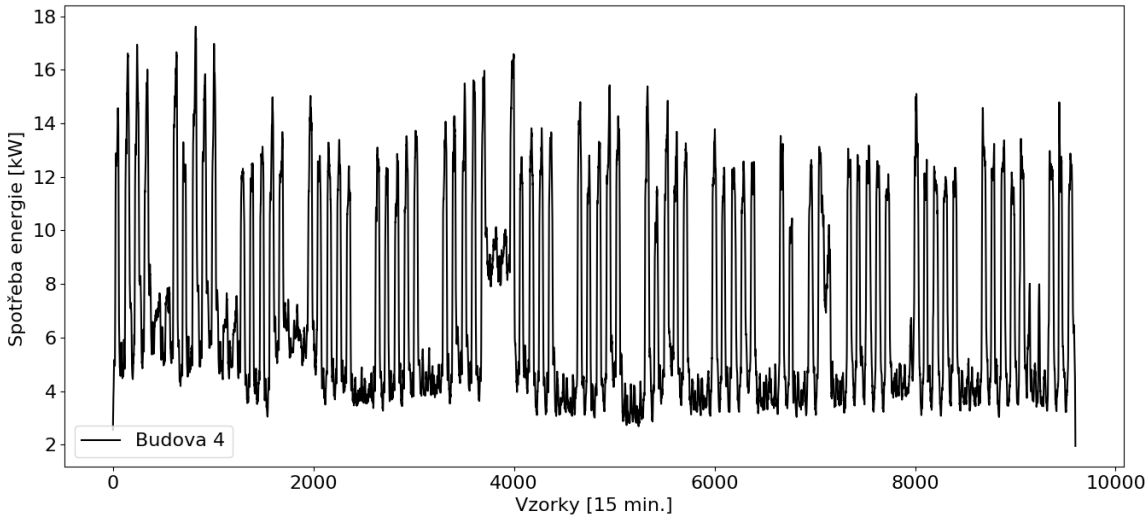
Obr. 23: Entropie učení (graf obsahuje celý interval dat, na rozdíl od Obr. 22, ke kterému se vztahuje) obsahuje dva neobvyklé skoky. První skok (přibližně vzorek 700) naznačuje první naučení na datech. Druhý skok (přibližně 9000) nastal po vyřazení učení, tedy po posledním vyznačeném neobvyklém stavu(Obr. 21).

Tři další detekované chyby však neodpovídají očekávání. Z výsledku entropie učení na Obr. 23 je však vidět, že entropie udělá náhlý veliký skok po přerušení učení. Z Obr. 21 lze vidět, že všechny týdny po poslední chybě (vzorky 9000 a dál) mají neobvykle velikou spotřebu energie oproti předešlým týdnům. Lze tedy konstatovat, že detekce posledních tří chyb vznikla změnou chování naměřených dat. V tomto vyjímečném případě lze buďto snížit predikci nebo zvýšit detekční limit chyby predikce (na Obr. 22 je zřejmé, že první tři očekávané chyby mají větší procentuální chybu oproti zbylým chybám).

10.3 Budova s méně stabilním průběhem spotřeby energie

Druhá budova (Obr. 24) má mnohem nestabilnější průběh spotřeby energie. U této budovy (4.) jsem předem neoznačoval potenciální neobvyklé stavy a nechal jsem detekci volný průběh. Zpětně jsem však vyřadil učení v

intervalech, které byly pro učení viditelně chybné - (3600 4100), (6800 7400), viz. Obr. 24,Obr. 25,Obr. 27.



Obr. 24: Spotřeba energie 100 dní budovy 4 s nevyznačenými neobvyklými stavy.

Zkusil jsem změnit parametry predikce na velmi krátkou – 4 vzorky dopředu (1 hodina) a trochu jsem pozměnil vstupní vektor do neuronové sítě (nastavení dle Tab 10,Tab 11,Tab 12). Z výsledků (Obr. 25,Obr. 26,Obr. 27,Obr. 28) je celkem patrné, že kratší predikce má velmi dobré výsledky i při méně stabilní spotřebě energie.

Nastavení parametrů dle Tab 10.

Neuronová síť	HONU (CNU)
Učící algoritmus	CG
Predikce	+4 vzorků
Trénovací okno	7 dnů (672 vzorků)
Přetrénovávání	1 den (96)
Epoch naučení	30
Epoch přeučování	30

Tab 10: Parametry neuronové sítě, predikce a učení

Vstupní vektor dle Tab 11.

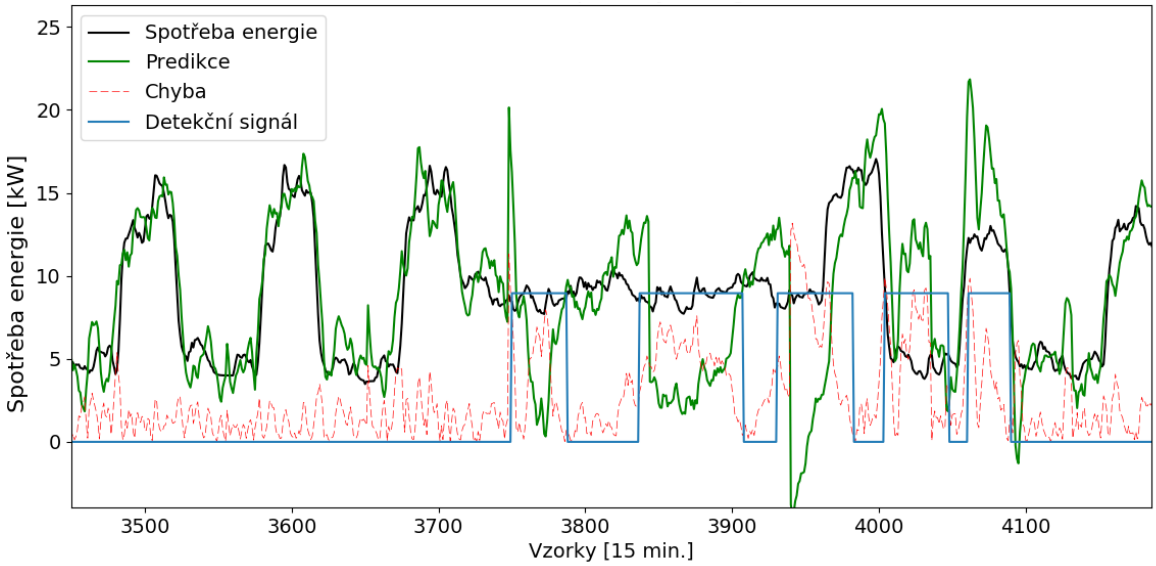
	Počet zpožděných hodnot	Mezera mezi hodnotami
Naměřené hodnoty	12	8
Dny	1	1
Čas	1	1

Tab 11: Vstupní vektor

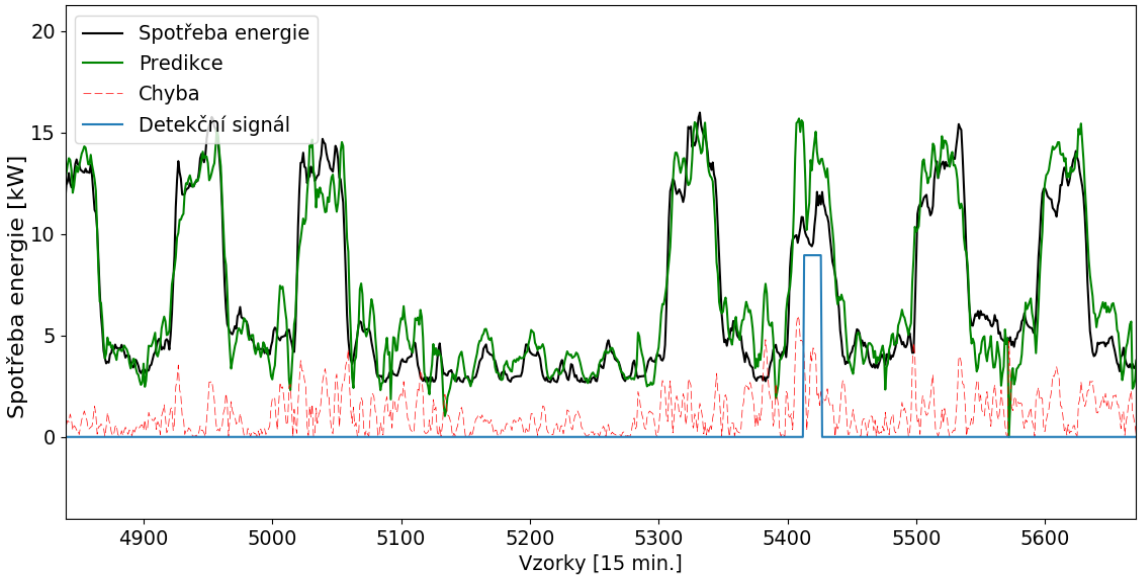
Detekci chyb dle Tab 12

Počet vzorků ze kterých se vytvoří průměrná chyba	Hodnota, vůči které se bude srovnávat průměrná chyba	Limit detekce chyby
16	14kW	20%

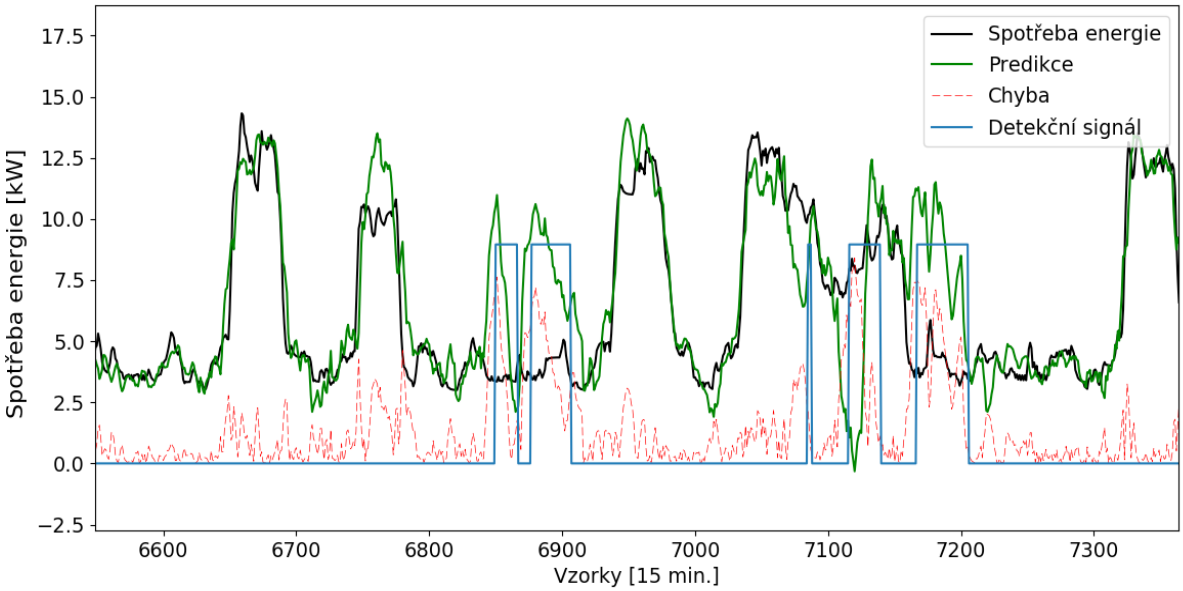
Tab 12: Parametry detekce chyb



Obr. 25: Predikce a detekce neobvyklých stavů budovy 4 (p=4,HONU – CNU). Byla detekována vyšší spotřeba o víkendu.

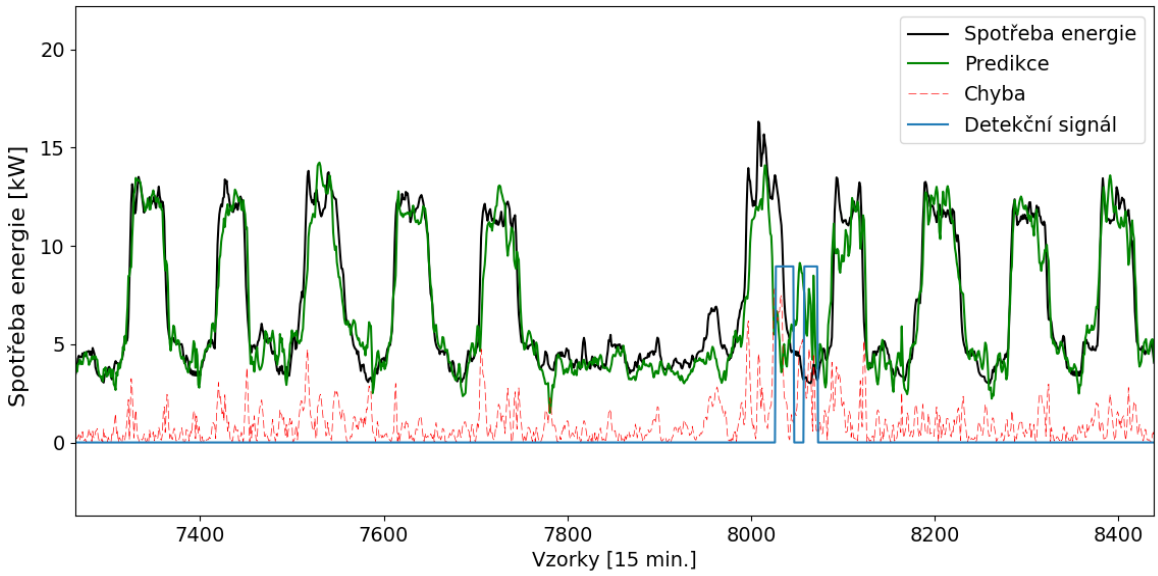


Obr. 26: Predikce a detekce neobvyklých stavů budovy 4 (p=4,HONU - CNU). Byla detekována neobvykle nižší spotřeba energie ve všední den.



Obr. 27: Predikce a detekce neobvyklých stavů budovy 4 (p=4,HONU - CNU). Byla detekována neobvykle nízká spotřeba ve všední den(vzorky 6800 až 6900) a neobvyklá spotřeba mezi čtvrtek a pátkem (vzorky 7100 až 7200)

Na Obr. 28 je vidět jak neobvyklý stav (vyšší spotřeba okolo vzorků 8000) nebyl detekován predikcí, ale tím že naměřené hodnoty byly prostě neobvyklé a jako součást vstupního vektoru způsobily špatnou predikci. Detekce chyby tedy přišla až po samotném neobvyklém stavu.



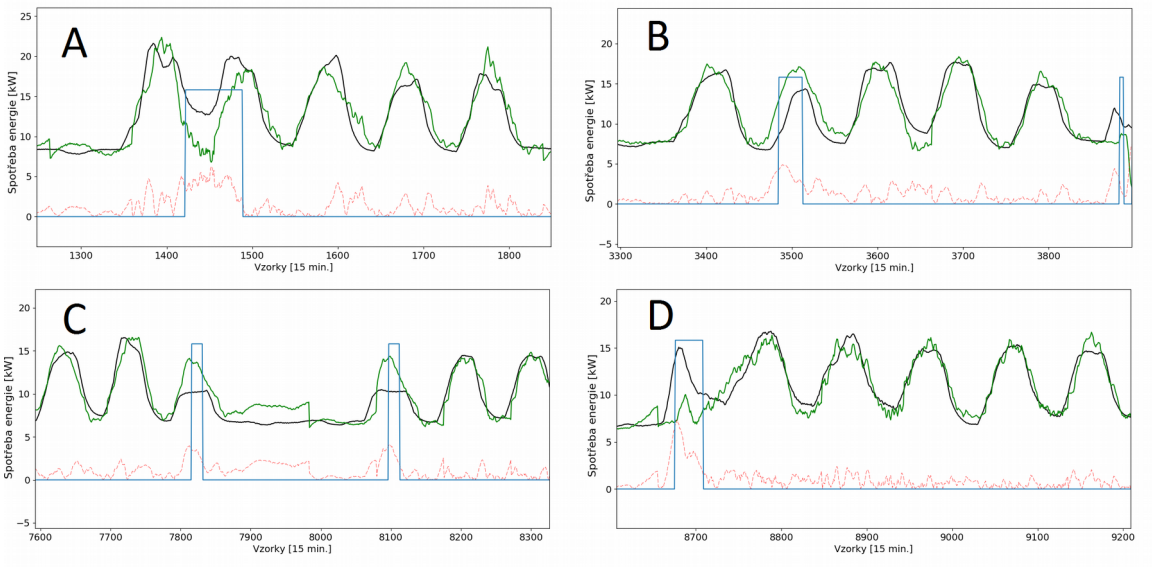
Obr. 28: Predikce a detekce neobvyklých stavů budovy 4 (p=4, HONU – CNU,CG).Byla detekována neobvykle vysoká spotřeba v pondělí. Zvláštností je, že detekce byla spuštěna chybou predikce, která nastala neobvyklými naměřenými hodnotami ve vstupním vektoru.

U budovy 4 nedošlo k žádným výpadkům energetické činnosti (např .Obr. 21, první označený neobvyklý stav). Data však obsahovala neobvyklé chování spotřeby energie. Ukázalo, že krátkodobá predikce je nejen vhodná na méně stabilní data, ale i na detekci neobvyklého chování dat, které neobsahují očividně chybné stavy jako jsou výpadky energetické činnosti.

10.4 Porovnání krátkodobé a dlouhodobé predikce

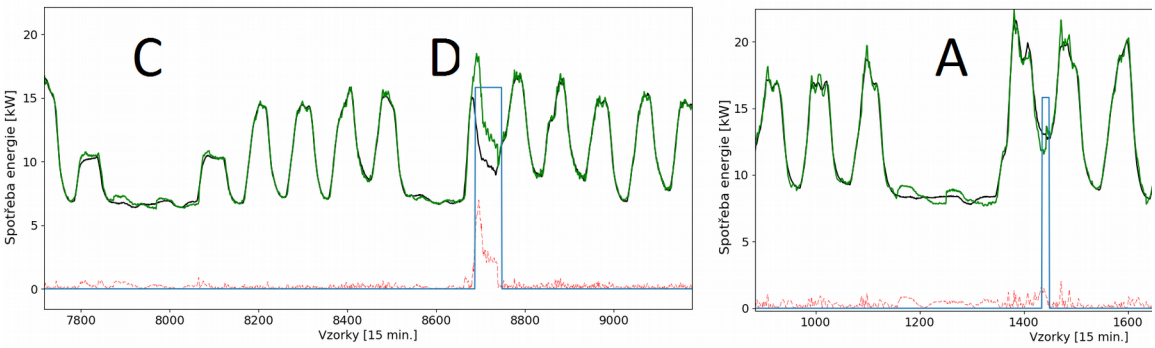
Nejvíce problematické v detekci neobvyklých stavů je určit, co má vlastně být detekováno jako neobvyklý stav. Tato volba záleží na uživateli mé aplikace. Na základě nastavení parametrů predikce ladícího nástroje je možné detekovat různé druhy neobvyklých stavů. Pro názornost předvedu rozdíl v detekci neobvyklých stavů při krátkodobé (p=3) a dlouhodobé (p=16 vzorků) predikci. Na Obr. 29 jsou výsledky detekce neobvyklých stavů budovy

2, mezi 3. a 120. dnem, při dlouhodobé predikci s MLP sítí a LM (vstupní vektor dle Tab 11, úprava dat dle 8.11 - coarse graining, radius = 5-10).



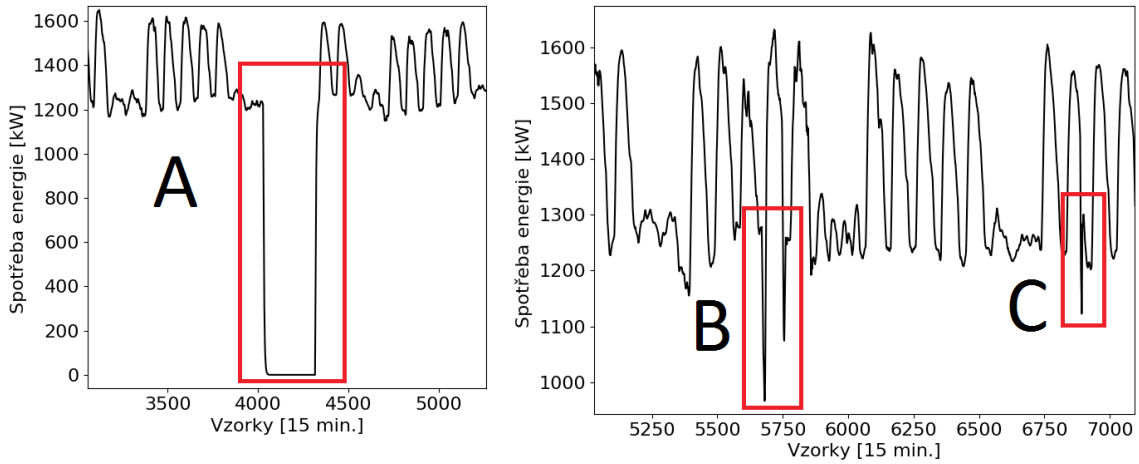
*Obr. 29: Detekce neobvyklých stavů budovy 2 (p=16, MLP-LM). Detekce **A** naznačuje výšší spotřebu energie přes noc. **B** naznačuje nižší spotřebu energie během všedního dne. **C** naznačuje nízkou spotřebu energie o Velikonocích. **D** naznačuje neobvyklou spotřebu energie o víkendu.*

Při krátkodobé predikci s HONU-CNU a CG (Obr. 30) byly chyby **D** a **A** v obou případech detekovány jako neobvyklé stavy. Na obrázku se stavem **C** jsou státní svátky a je vidět že při dlouhodobé predikci se se svátkami počítá jako s neobvyklým stavem, zatímco při velmi krátkodobé(p=3) predikci se se svátkami počítá jako s obvyklými stavy. Avšak při predikci p=5 se už se svátkami začíná počítat jako s neobvyklým stavem.



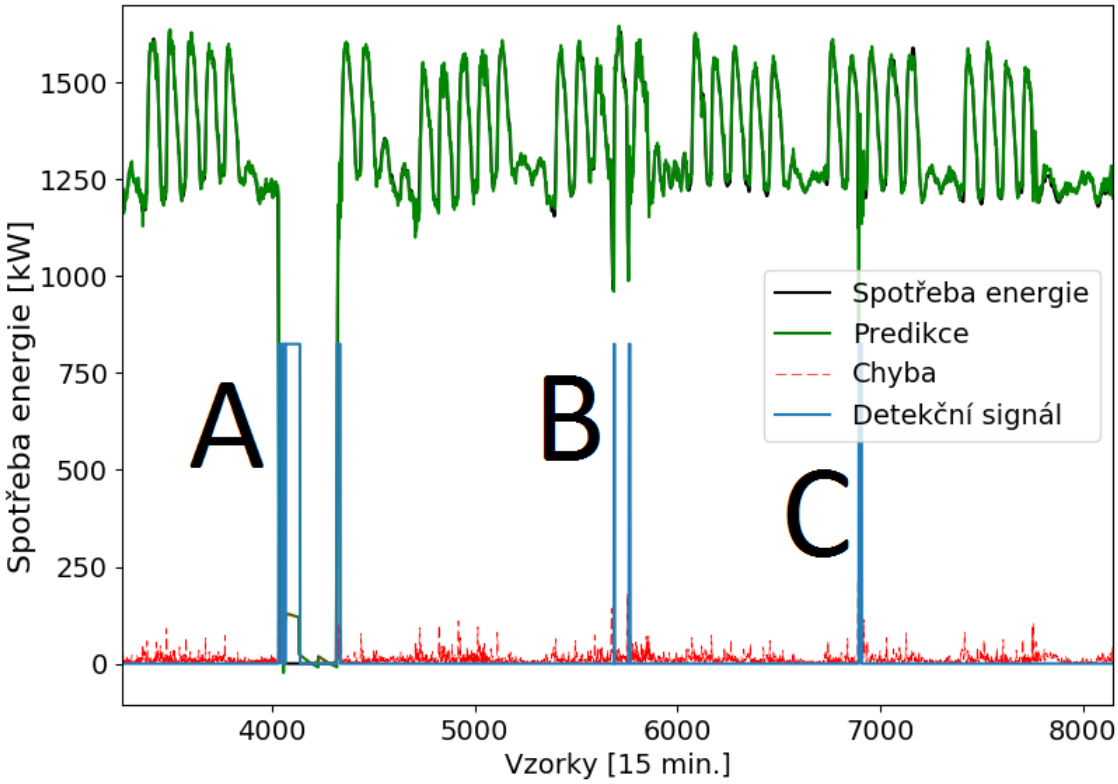
*Obr. 30: Detekce neobvyklých stavů budovy 2 (p=3 HONU-CNU,CG). Oproti detekcím z Obr. 29 jsou při krátkodobé predikci detekovány pouze stavy **D** a **A**.*

V datech se poměrně často objevují stavy, které připomínají náhlý výpadek energetické činnosti (Obr. 31).



*Obr. 31: Budova 9 s vyznačenými výpadky energetické činnosti **A,B,C**.*

Je-li požadavek na detekci pouze takovýchto stavů, pak je výhodné použít velmi krátkou predikci (p=3 až p=5). Detekují se totiž především takovéto stavy a



Obr. 32: Detekce náhlých výpadků energetické činnosti A,B,C vyznačených z naměřených dat (Obr. 31).

to i u budov s méně stabilní spotřebou energie. Na Obr. 32 jsou výsledky detekce neobvyklých stavů z Obr. 31 s HONU-CNU-CG. Výsledky jsou velmi uspokojivé, neboť byly detekovány všechny výpadky energetické činnosti bez jakýchkoliv jiných detekcí a to i přes to, že se neuronová síť učila i na těchto stavech. Závěrem je potřeba zdůraznit, že pro získání uspokojivých výsledků je potřeba stanovit, jaké neobvyklé stavy detekovat, dále pak naladit neuronové sítě a v poslední řade je důležité správně nastavit vyhodnocení chyb predikce vůči naměřeným hodnotám.

10.5 Možnost nepřetrénovávání neuronové sítě

Je možné neuronovou síť naučit pro jeden daný interval a dále již neučit, avšak musí být dodrženo několik podmínek:

- Krátkodobější predikce (nejlépe p=5)
- Střední nebo vysoké vyhlazení dat (radius = 3 až 10 dle 8.11)

- Data, které mají přibližně konstantní rozsah spotřeby energie,

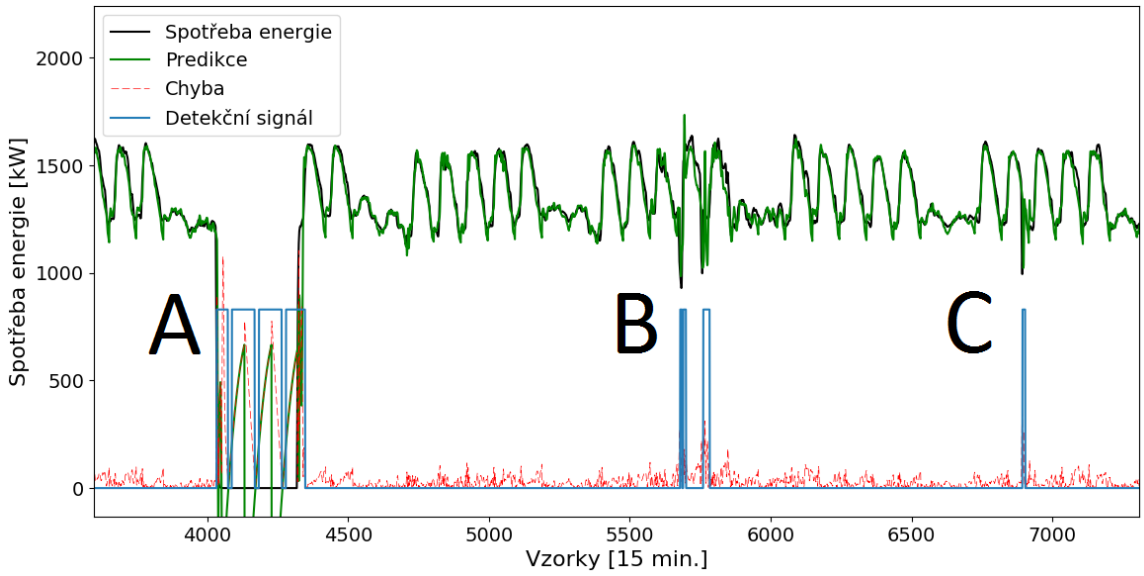
V datech ENERGOCENTRUM Plus je několik budov, na které lze aplikovat tuto možnost. Tato metoda se zdá být celkem uspokojivá, avšak je dle mého názoru nespolehlivá. Uvedu zde příklady dvou uspokojivých výsledků a jednoho výsledku, který dokazuje nespolehlivost této metody.

Pro porovnání jsem detekoval opět budovu 9 z předchozí podkapitoly 10.4. Data tedy odpovídají těm na Obr. 31. Vstupní vektor jsem nastavil dle Tab 13.

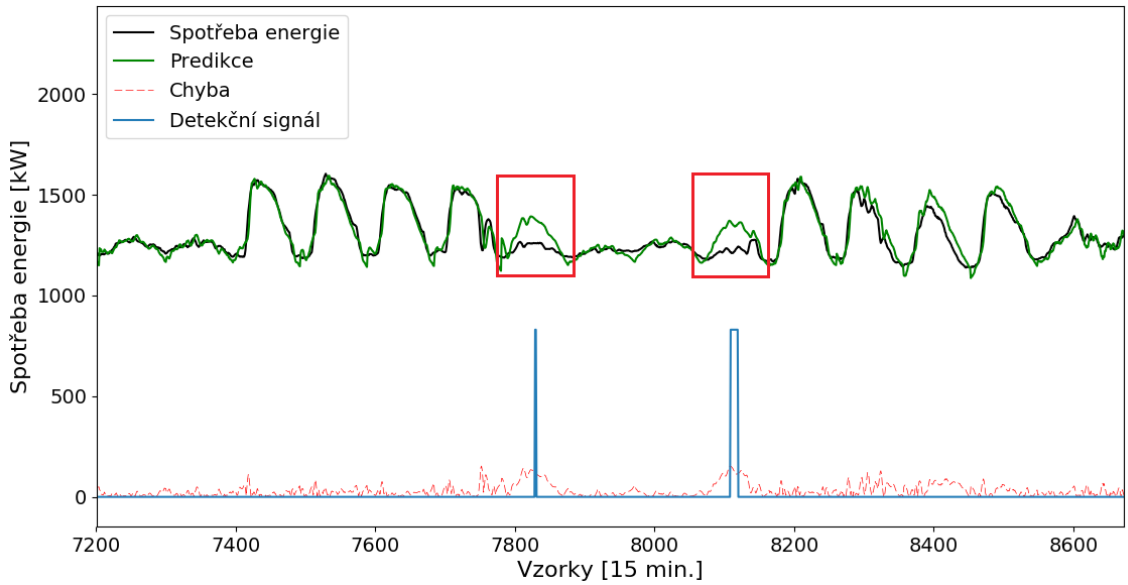
	Počet zpožděných hodnot	Mezera mezi hodnotami
Naměřené hodnoty	8	3
Dny	1	1
Čas	1	1

Tab 13: Vstupní vektor do neuronové sítě MLP

Pro změnu jsem použil MLP-LM z toho důvodu, že v tomto případě dává při krátkodobé predikci lepší detekci neobvyklých stavů než HONU-CNU-CG. Predikci jsem nastavil na p=5 a vyhlazení dat radius=3 (Při větším vyhlazení začnou být náhlé výpadky energetické činnosti hůře detekovatelné). Zbylé parametry dle Tab 6. Učící proces probíhal do vzorku 2000. Výsledky na Obr. 33 naznačují zachycení všech výpadku energetické činnosti.



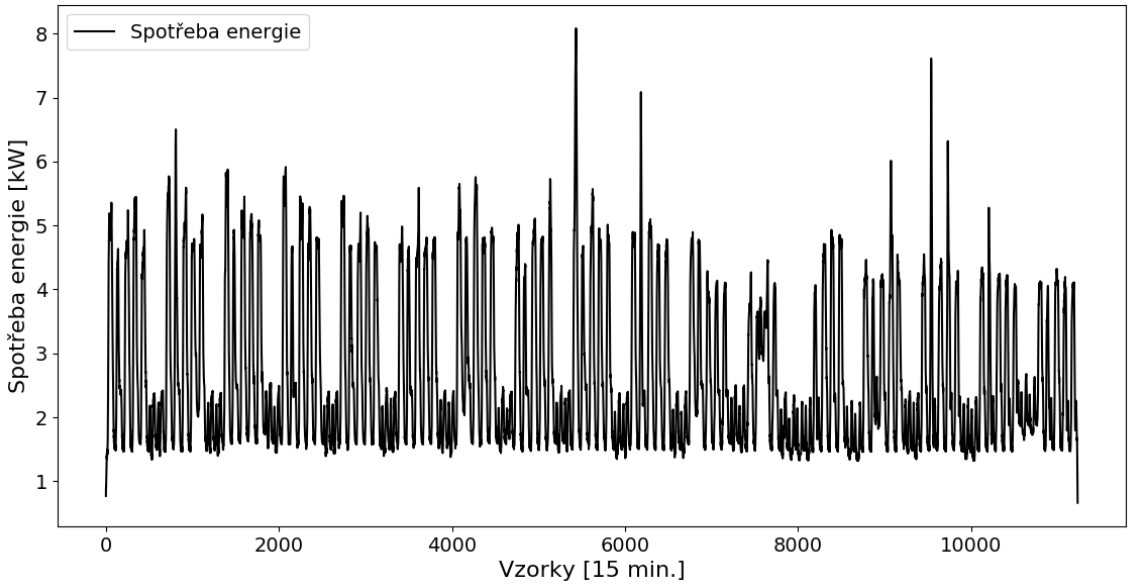
Obr. 33: Detekce náhlých výpadků energetické činnosti A,B,C vyznačených z naměřených dat (Obr. 31). Použití sítě MLP – LM, p=5. Neuronová síť se nepřetrénovávala



Obr. 34: Detekce svátků budovy 9 (data z Obr. 31). Neuronová síť se nepřetrénovávala.

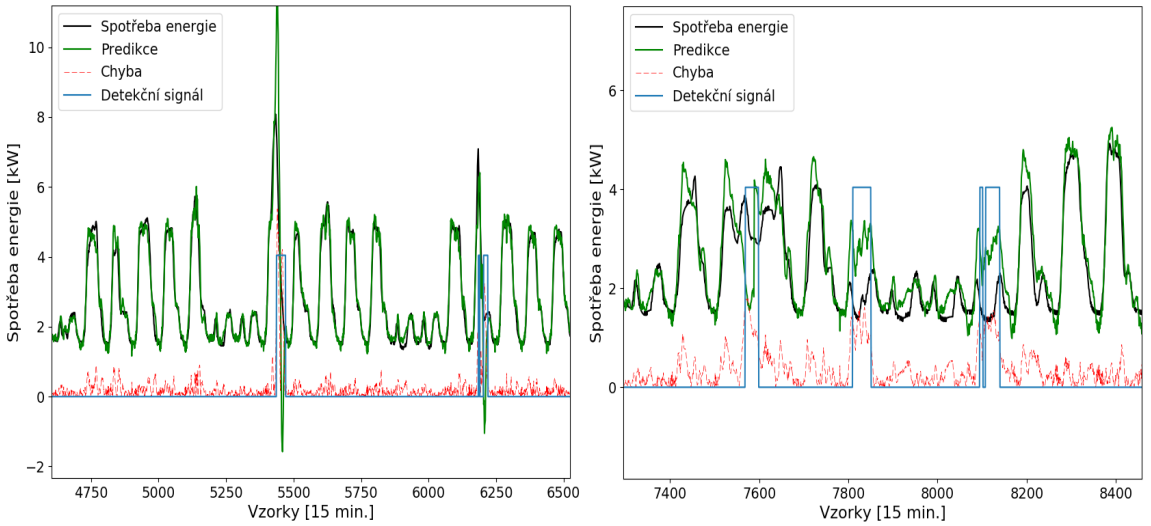
Jelikož jsem zvolil predikci p=5 (v kapitole 10.4 jsem zvolil p=3), začaly se projevovat i jiné neobvyklé stavy, které se detekovaly, a to svátky (Obr. 34).

Další budova (8. data), na které jsem ověřil možnost nepřetrénovávat neuronovou síť neobsahovala žádné náhlé výpadky energetické činnosti a proto jsem zvedl radius vyhlazování dat na hodnotu 6.



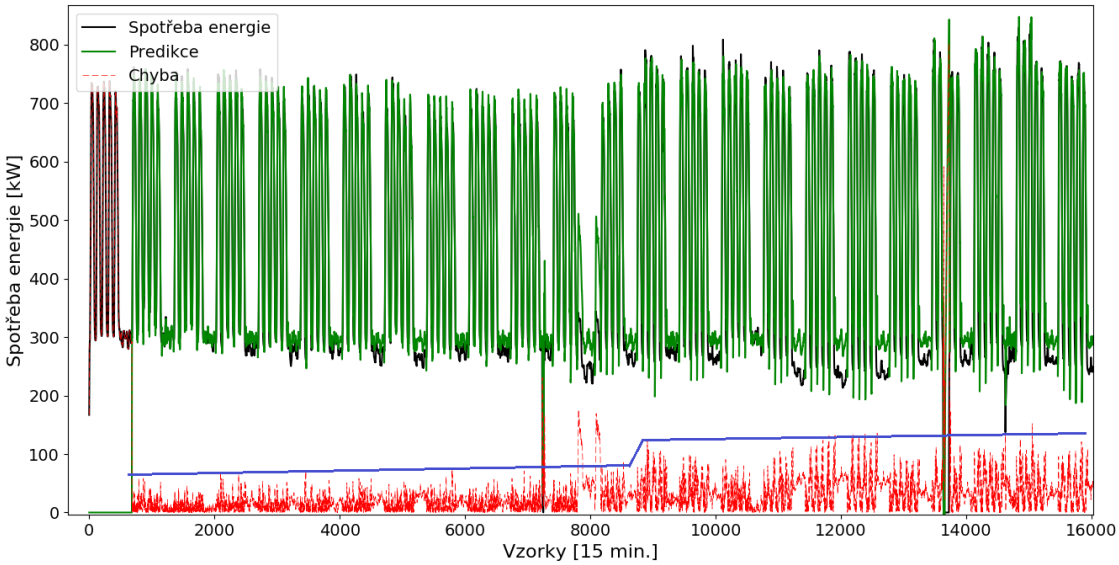
Obr. 35: Spotřeba energie budovy 8.

Predikci jsem ponechal na p=5. Zvolil jsem HONU-CNU-CG, která měla překvapivě u předchozí budovy horší výsledky než MLP-LM a najednou má mnohem lepší výsledky při stejném nastavení. Výsledky (Obr. 36) jsou velmi dobré, neboť nedošlo k žádné falešné detekci.



Obr. 36: Detekce neobvyklých stavů při predikci spotřeby energie z budovy 8 bez přetrénovávání neuronové sítě, nastavení HONU-CNU-CG, p=5. Graf nalevo naznačuje detekci neobvykle vysoké spotřeby energie. V grafu napravo je detekována neobvyklá spotřeba přes noc a dále pak nízká spotřeba o svátcích.

Na poslední budově 10 (nejstabilnější data viz. Obr. 21) předvedu hlavní nedostatek této metody.



Obr. 37: Predikce spotřeby energie budovy 10. Okolo vzorku 8000 nastane změna dynamiky chování dat, což způsobilo větší chybovost predikce neuronovou sítí, která se na datech neučí. Modrá čára je ilustrativní a vyjadřuje náhlou změnu chování dat

Ukázalo se, že ačkoliv data jsou velmi stabilní, v určitém okamžiku se změní chování spotřeby energie a neuronová síť, která se nepřetrénovává, začne predikovat s vyšší chybovostí (viz modrá čára, Obr. 37). Zvláštností je, že budovy u předchozích dvou příkladů měli oproti budově 10 celkem kolísavou spotřebu energie, přesto však chyba predikce byla konstantní v celém rozsahu výpočtu. U budovy 10 je kolísavost minimální (Obr. 37). Stačila však jedna změna v dynamice dat a chybovost predikce (u vzorku cca 9000, Obr. 37) se zvětšila. **Nespolehlivost této metody tedy vidím v možnosti změny dynamiky chování dat**.

11 Vyhodnocení výsledků

V této kapitole shrnu výsledky dosažených pomocí ladícího nástroje ve vztahu k datům z ENERGOCENTRUM Plus a uvedu zde teoretické poznatky, které jsem získal při ladění neuronových síti k dosažení uspokojivých výsledků .

11.1 Neuronové modely a učící algoritmy

Nejlépe naučitelné modely pro delší predikci (p=16) jsou jednoznačně MLP síť s LM učícím algoritmem a HONU s CNU neuronem a s LM a CG učícími algoritmy. Naopak při krátkodobé predikci (p=3 až p=5) si vedou všechny modely velmi dobře včetně krokových učících algoritmů, avšak nejlépe si vedou opět výše zmíněné typy (MLP-LM, HONU-CNU-CG, LM). Nejlepší metodika ladění je testovat dlouhodobou predikci s HONU CNU neuronem, jehož výpočet je velmi rychlý a po naladění zkusit MLP síť, zda si nevede lépe. U krátkodobé predikce je výhodné začínat s HONU LNU a postupně navyšovat řád (ze zkušenosti mohu říct, že pokud LNU nedává uspokojivé výsledky, lze rovnou přeskočit na CNU) a až v poslední řadě zkusit MLP.

11.2 Data od ENERGOCENTRUM Plus

Data od ENERGOCENTRUM Plus jsem v některých případech vyhodnotil jako nedostačující. Přibližně u třetiny budov je velmi těžké provádět detekci, jelikož je zřejmá absence dodatečné vstupní veličiny do neuronové sítě, která lépe postihuje chování dat, tedy spotřebu energie. Lze v podstatě říct, že spotřeba energie každé budovy představuje odlišnou soustavu a jediné co spojuje všechny tyto soustavy jsou mnou zvolené vstupní veličiny. U těchto dat je dále velmi problematické, že kvůli nestabilní spotřebě energie a nedostatku vstupních veličin nelze určit referenční data z kterých by vyplývala predikce. Kvalitu predikce však lze ovlivnit vyhlazováním dat klouzavým průměrem a zkracováním vzdálenosti predikce (vyhlazování dat a zkracování predikce má vliv na to, jaké neobvyklé stavy se budou detekovat viz podkapitola 11.3). Ve většině případů však lze získat uspokojivé výsledky pokud se zvolí správné parametry a vyřadí se učení neuronové sítě na datech s neobvyklými stavy. Data jsou ve své vlastní struktuře tak odlišná, že je mnohdy potřeba otestovat více kombinací nastavení vstupního vektoru pro dosažení nejlepších výsledků (nejvíce se mi osvědčilo nastavení Tab 13 a pak kombinace z Tab 7 a Tab 11).

11.3 Neobvyklé stavy

Velmi problematické je určit, co v datech je neobvyklý stav. Výpadek energetické činnosti (např. Obr. 33) a spotřeba energie, která se vymyká obvyklému chování daných dat (např. Obr. 30-D) jsou zřejmými příklady neobvyklých stavů a jsou ve většině případů detekovány při různých nastavení neuronové sítě. ‚Ostatní' stavy zachovávají dynamiku spotřeby energie, avšak je zřejmé, že se vymykají rutině daných dat (např.Obr. 30-C, Obr. 29-C -svátky, nebo Obr. 29-B) a je pro ně charakteristické, že jsou detekovány jen pro některé nastavení neuronové sítě. Je tedy důležité si předem stanovit, co je neobvyklý stav a co není.

Pro detekci zřejmých neobvyklých stavů (např. výpadek energetické činnosti) je nutné nevyhlazovat data příliš (dle 8.11, max. radius=5, doporučený radius=3). Pro dlouhodobou predikci je chybovost predikce větší, proto je výhodné vyhladit data více. Obecně vzato lze říct, že vyhlazování dat zlepšuje predikci, ale zároveň „opravuje" zřejmé neobvyklé stavy. Moje doporučení je pro predikci p=3 až p=5 mít vyhlazení dat s radiusem 3 až 6. Pro dlouhodobou predikci predikci bych volil radius od 4 až do 10.

11.4 Dlouhodobá a krátkodobá predikce

Z hlediska dat od ENERGOCENTRUM Plus považuji krátkodobou predikci 3 až 5 vzorků dopředu a dlouhodobou predikci až 16 vzorků dopředu. Toto rozdělení má svoje opodstatnění. Při krátkodobé predikci se s největší pravděpodobností detekují jasné neobvyklé stavy (dle kapitoly 11.3). Predikce 5 vzorků dopředu je hraniční a při tomto nastavení se začínají detekovat i ‚ostatní' stavy (dle kapitoly 11.3, například nižší energetická činnost v období svátků). Při dlouhodobé predikci (až p=16) se detekuje v podstatě vše, co je mimo rutinu (referenční data). Při tomto nastavení se však mohou detekovat i normální stavy jako neobvyklé. Predikci dál než p=16 nedoporučuji, protože začíná vznikat vysoká chyba mezi vypočtenou a naměřenou hodnotou, což způsobuje častější falešnou detekci.

Celkově mi přišla jako nejlepší volba krátkodobá predikce (5 vzorků), která s jistotou detekuje zřejmé neobvyklé stavy a je s ní možné detekovat i ‚ostatní' stavy. Je nutné dodat, že krátkodobá predikce je nejlepší při volbě HONU-CNU-CG (avšak není to pravidlo, neboť se ukázalo že v některých případech mělo LNU a MLP lepší výsledky). Při použití MLP-LM není vždy zaručena detekce všech jasně neobvyklých stavů . Nicméně je lepší otestovat oba modely. Ukázalo se totiž, že pro dvě různá data si oba modely vedou rozdílně (při stejných nastavených parametrech). Je-li požadavek detekovat jakýkoliv podezřelý stav a nevadí detekce i normálních stavů, pak doporučuji dlouhodobou predikci s volbou MLP-LM a nebo pro rychlý výpočet HONU-CNU-CG.

11.5 Učení na chybných datech

Velmi důležité opatření pro správnou detekce neobvyklých stavů je prevence učení neuronové sítě na chybných datech. Aplikace má nástroj na toto opatření, avšak pro správnou prevenci je nutný určitý postup. Uživatel může buďto vyřadit učení v chybných intervalech manuálně, nebo automaticky v intervalech detekovaných neobvyklých stavů. Automatický způsob je samozřejmě pohodlnější a časově nenáročnější, avšak méně přesnější.

Pro správný postup automatického způsobu doporučují nejdříve pomocí krátkodobé predikce nalézt jasné neobvyklé stavy a na základě těchto nálezů vygenerovat intervaly pro přerušení. V dalším kroku již lze pokračovat v dalším ladění při stejných intervalech (popřípadě aktualizovat intervaly). Důvod tohoto postupu je, že při větším počtu detekovaných neobvyklých stavů se vyřadí příliš intervalů a důsledkem toho vzniknou falešné detekce, neboť se neuronová síť nepřetrénovává. Dále je výhodné při automatickém vygenerování intervalů smazat intervaly na začátku (do cca 1000 až 2000 vzorků) a to z toho důvodu, že síť není ještě plně naučená a predikce tedy způsobuje chyby.

11.6 Nepřetrénovávání neuronové sítě

Neuronovou síť je možné jednou naučit a dále nepřetrénovávát. Tím se lze vyhnout mnoha nesnázím, jako třeba problému učení na chybných datech. Je však nutné dodržet několik velmi omezujících podmínek na data dle podkapitoly 10.5. Jsou-li splněny tyto podmínky, pak jsou výsledky u některých dat ENERGOCENTRUM Plus velmi dobré (podkapitola 10.5). Je překvapivé, že tento způsob má lepší výsledky u budov s méně stabilní spotřebou energie než u budovy 10 (Obr. 21), jejíž spotřebu energie považuji za nejstabilnější. Další výhodou je extrémně rychlý výpočet.

Celkově vzato je tento způsob predikce velmi výhodný, avšak je potřeba pro něj najít správná data.

12 Závěr

Cílem této diplomové práce bylo navrhnout a naprogramovat ladící nástroj pro predikci dat energetických budov a detekci neobvyklých stavů neuronovými sítěmi. Základním požadavkem bylo v datech poskytnutých ENERGOCENTRUM Plus detekovat neobvyklé stavy.

V Pythonu jsem naprogramoval aplikaci, ve které je možné si pro predikci dat vybrat neuronový model (HONU, MLP), učící algoritmus (Konjugovaný Gradient (pouze HONU), Levenberg-Marquardt, Gradient Descent, Normalizovaný Gradient Descent), a nastavit dodatečné parametry. Součástí aplikace jsou dále dva moduly. První je na vyhodnocování chyb predikce s možností detekce neobvyklých stavů. Druhý modul je na detekci učení na chybných datech metodou Learning Entropy for Novelty Detection s možností naladění této metody a možností vyřazení přetrénovávání neuronové sítě v definovaných intervalech. Nad rámec diplomové práce jsem dále naprogramoval přídavný modul, ve kterém je možné provádět predikci libovolných dat. Tento přídavný modul lze tedy použít i na jiná data než od ENERGOCENTRUM Plus s libovolnými vstupními veličinami a přitom využívat všechny možnosti, které aplikace nabízí.

Pomocí mé aplikace lze dosáhnout uspokojivých výsledků, nicméně je potřeba, aby uživatel aplikace správně naladil všechny nezbytné parametry pro predikci dat a detekci neobvyklých stavů a postupoval při tom tak, jak jsem naznačil v kapitolách 9,10,11. Data od ENERGOCENTRUM Plus nejsou v některých případech vhodná, a to buď z důvodu nedostatku vstupních veličin, na kterých jsou výstupní data závislá, nebo kvůli velmi malému rozlišení dat. Ukázalo se však, že při predikci přibližně hodinu dopředu (odpovídá 4 vzorkům) lze s velikou jistotou detekovat zřejmé neobvyklé stavy, jako například výpadky energetické činnosti, pro většinu dat. Naopak při predikci až 4 hodiny dopředu (odpovídá 16 vzorkům) lze detekovat více druhů neobvyklých stavů, avšak ne pro všechny data jsou výsledky dostatečně uspokojivé.

Aplikace je navržená tak, aby se dle potřeby dala modifikovat. Je tedy možné aplikaci do budoucna rozšiřovat a to jak z hlediska nových algoritmů učení, tak i nových metod vyhodnocování chyb a detekce. V navazujících pracích by bylo vhodné vyzkoušet ladící program i na jiných datech energetických budov s více vstupními veličinami. Výhodné by též bylo vyvinout aplikaci, která automaticky zpracovává data od ENERGOCENTRUM Plus či jiná energetická data a automaticky detekuje neobvyklé stavy. Při testování dat od ENERGOCENTRUM Plus pomocí mé aplikace se ukázalo, že naladění neuronové sítě, ošetření intervalů s neobvyklými stavy, na kterých se neuronová síť může chybně učit a celková odlišnost dynamiky chování spotřeby energie různých budov je velmi komplexní problém. Vývoj takovéto aplikace by sice byla velká výzva, avšak dle mého názoru by takováto aplikace byla velmi žádaná.

13 Použité zdroje

[1] J. Čepela, *Studie využití neuronových sítí pro predikci spotřeby komplexu budov*. ČVUT FS, Odbor automatického řízení a inženýrské informatik, 2014.

[2] „Reference | Energocentrum.cz". [Online]. Dostupné z: http://www.energocentrum.cz/reference/. [Viděno: 08-čer-2017].

[3] H. R. Khosravani, „Energies | Free Full-Text | A Comparison of Energy Consumption Prediction Models Based on Neural Networks of a Bioclimatic Building", 15 October 20015. [Online]. Dostupné z: http://www.mdpi.com/1996-1073/9/1/57. [Viděno: 14-dub-2017].

[4] „Building heating load estimation using artificial neural networks", *ResearchGate*. [Online]. Dostupné z: https://www.researchgate.net/publication/228903198_Building_heating_load_estimation_using_artificial_neural_networks. [Viděno: 14-dub-2017].

[5] B. B. Ekici a U. T. Aksoy, „Prediction of building energy consumption by using artificial neural networks - Semantic Scholar", 2009. [Online]. Dostupné z: /paper/Prediction-of-building-energy-consumption-by-using-Ekici-Aksoy/a7531283bd524aca4fe29a352150b123e9b33534. [Viděno: 14-dub-2017].

[6] Y. Cheng-wen a Y. Jian, „Application of ANN for the prediction of building energy consumption at different climate zones with HDD and CDD", in *2010 2nd International Conference on Future Computer and Communication*, 2010, roč. 3, s. V3-286-V3-289.

[7] L. Kracík, „Počasí neovlivníte, ale můžete s ním obchodovat přes deriváty", *Měšec.cz*. [Online]. Dostupné z: https://trhy.mesec.cz/clanky/jak-funguji-derivaty-spojene-s-pocasim/. [Viděno: 14-dub-2017].

[8] A. Kalogirou, „Artificial neural networks for the prediction of the energy consumption of a passive solar building", kvě-2000. [Online]. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0360544299000869. [Viděno: 14-dub-2017].

[9] I. Bukovský, „Higher Order Neurons and Supervised Learning for Prediction, Control, and Novelty Detection". TOHOKU UNIVERSITY Tohoku University Graduate School of Medicine Dpt. of Radiological Imaging and Informatics Sendai, Japan CZECH TECHNICAL UNIVERSITY IN PRAGUE FME, Dpt. of Instrumentation and Control Engineering Prague, Czech Rep, 25-srp-2016.

[10] I. Bukovský, P. Beneš, M. Veselý, a J. Kalivoda, „Poznatky z výzkumu neuro-regulátorů a z laboratorní praxe". Automatizace, regulace a procesy 2016, 22-dub-2017.

[11] I. Bukovsky a N. Homma, „An Approach to Stable Gradient-Descent Adaptation of Higher Order Neural Units", *IEEE Trans. Neural Netw. Learn. Syst.*, roč. PP, č. 99, s. 1–13, 2016.

[12] M. Gupta, L. Jin, a N. Homma, *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*. John Wiley & Sons, 2004.

[13] „Neural networks and deep learning". [Online]. Dostupné z: http://neuralnetworksanddeeplearning.com/chap2.html. [Viděno: 31-bře-2017].

[14] I. Bukovsky, „Dynamický Backpropagation", *Automatizace*, roč. 52, No. 10, s. p.586-590, led. 2010.

[15] R. Martínek, „Využití adaptivních algoritmů LMS a RLS v oblasti adaptivního potlačování šumu a rušení", s. 3, bře. 2013.

[16] J. J. Moré, „The Levenberg-Marquardt algorithm: Implementation and theory", in *Numerical Analysis*, Springer, Berlin, Heidelberg, 1978, s. 105–116.

[17] „Levenberg-Marquardt backpropagation - MATLAB trainlm". [Online]. Dostupné z: https://www.mathworks.com/help/nnet/ref/trainlm.html. [Viděno: 02-dub-2017].

[18] „Hessian matrix of scalar function - MATLAB hessian". [Online]. Dostupné z: https://www.mathworks.com/help/symbolic/hessian.html#buiej1q-2. [Viděno: 04-dub-2017].

[19] K. LEVENBERG, „A METHOD FOR THE SOLUTION OF CERTAIN NON-LINEAR PROBLEMS IN LEAST SQUARES", *Q. Appl. Math.*, roč. 2, č. 2, s. 164–168, 1944.

[20] M. R. Hestenes a E. Stiefel, *Methods of conjugate gradients for solving linear systems*. National Bureau of Standards, 1952.

[21] J. R. Shewchuk, „An Introduction to the Conjugate Gradient Method Without the Agonizing Pain", Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

[22] P. de Chazal, J. Tapson, a A. van Schaik, „A comparison of extreme learning machines and back-propagation trained feed-forward networks processing the mnist database", in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, s. 2165–2168.

[23] J. Tang, C. Deng, a G. B. Huang, „Extreme Learning Machine for Multilayer Perceptron", *IEEE Trans. Neural Netw. Learn. Syst.*, roč. 27, č. 4, s. 809–821, dub. 2016.

[24] H. Guang-bin, Z. Qin-yu, a S. Chee-kheong, „Extreme learning machine: Theory and applications", in *ScienceDirect*, 2006, s. 490–493.

[25] E. Cambria *et al.*, „Extreme Learning Machines [Trends & Controversies]", *IEEE Intelligent Systems*, roč. 28, č. 6, s. 30–59, 2013.

[26] H. Guang-bin, „Introduction to Extreme Learning Machine". National University of Singapore, 2006.

[27] M. F. Smith, *Software Prototyping: Adoption, Practice, and Management*. London ; New York: Mcgraw Hill Book Co Ltd, 1991.

[28] „What is Matlab". [Online]. Dostupné z: http://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm. [Viděno: 30-bře-2017].

[29] „MathWorks - Makers of MATLAB and Simulink". [Online]. Dostupné z: https://www.mathworks.com/. [Viděno: 04-kvě-2017].

[30] „The Julia Language". [Online]. Dostupné z: https://julialang.org/. [Viděno: 04-kvě-2017].

[31] „Welcome to Python.org", *Python.org*. [Online]. Dostupné z: https://www.python.org/. [Viděno: 04-kvě-2017].

[32] „Kivy: Cross-platform Python Framework for NUI Development". [Online]. Dostupné z: https://kivy.org/#home. [Viděno: 04-kvě-2017].

[33] „Rok 2016 - meteorologické statistiky". [Online]. Dostupné z: http://www.meteo.jankovic.cz/zaznamy/rok-2016/. [Viděno: 06-dub-2017].

[34] I. Bukovsky a C. Oswald, „Case Study of Learning Entropy for Adaptive Novelty Detection in Solid-Fuel Combustion Control", in *Intelligent Systems in Cybernetics and Automation Theory*, Springer, Cham, 2015, s. 247–257.

[35] I. Bukovsky, *Bukovsky, I.: Extended Dynamic Neural Architectures HONNU with Minimum Number of Neural Parameters for Evaluation of Nonlinear Dynamic Systems (in Czech)*. .

[36] I. Bukovsky, N. Homma, L. Smetana, R. Rodriguez, M. Mironovova, a S. Vrana, „Quadratic neural unit is a good compromise between linear models and neural networks for industrial applications", in *2010 9th IEEE International Conference on Cognitive Informatics (ICCI)*, 2010, s. 556–560.

14 Příloha

Seznam přílohy:

1. Instrukce k instalaci aplikace
2. Instrukce k používání dat
3. Zdrojový kód
4. CD s prací v elektronické podobě, aplikací a obrázky

14.1 Instalace aplikace

Postup instalace:

1. instalace Pythonu 2.7 (testováno na 2.7.12) z www.python.org
   - aktualizace Python nástrojů příkazem v Command Prompt:
   - **python -m pip install --upgrade pip wheel setuptools**
2. Instalace balíčků příkazem v Command Prompt:
   - **python -m pip install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew**
3. Instalace balíčku gstreamer příkazem v Command Prompt:
   - **python -m pip install kivy.deps.gstreamer**
   - Pokud se balíček nainstaluje, pokračujte bodem 6
   - Pokud se balíček nenainstaluje, pokračujte bodem 4 (manuální instalace gstreamer)
4. Stažení balíčku gstreamer pro správnou verzi Pythonu i Windows
   - https://kivy.org/downloads/packages/simple/kivy-deps-gstreamer/
5. Instalace balíčku gstreamer příkazem v Command Prompt (nutné být ve stejné složce jako soubor)
   - **python -m pip install kivy.deps.gstreamer-0.1.7-cp27-cp27m-win32.whl** (název musí odpovídat správné verzi)
6. Instalace kivy
   - **python -m pip install kivy**

7. Instalace numerické a grafické knihovny (numpy, matplotlib)
   - **python -m pip install matplotlib** (automaticky by se měla nainstalovat i knihovna numpy)
   - **python -m pip install numpy**
8. Instalace grafu matplotlib do kivy příkazy v Command Prompt
   - cd C:\Python27\Scripts
   - **garden install matplotlib**
   - **garden install graph** (s největší pravděpodobností není potřeba)

14.2 Instrukce k datům

Data se vkládájí do hlavní složky (se souborem main.py). Naměřené hodnoty musí být uloženy ve formátu txt, kde sloupce představují druhy dat a řádky vzorky dat (data jsou oddělené mezerou). Jméno souboru je **‚ENE_data.txt'**. Vzhledem k tomu, jak je aplikace naprogramována (pro ENERGOCENTRUM Plus) je vhodné, aby každý den měl 96 vzorků(ve skriptu main.py je tato proměná pod **self.sampling**) a především aby každý den měl stejný počet vzorků **(je proto nutné ošetřit jevy jako přechody na zimní a letní časy, kde den má den 97 a 95 vzorků (pokud nedojde k ošetření, pak v intervalu mezi změnou na letní čas a změnou na zimní čas bude vše posunuto o 1 vzorek). Doporučuji u budoucích dat smazat a přidat jeden vzorek u těchto dvou dnů)**.

Soubor se vstupní veličinou rozlišující všední dny od víkendů se jmenuje **‚log_days.txt',** se vstupní veličinou času **‚time.txt'** a se vstupní veličinou aproximované teploty **‚weather.txt'**. Aby všechny tři soubory spolu byly kompatibilní, je nutné, aby měli stejný počet řadků a tyto řadky spolu korespondovaly.

Ve složce ‚evaluation' jsou data z výpočtů spolu s dvěma skripty, které slouží k zobrazovaní právě těchto dat. Tyto skripty se spouští z programu.

Ve složce ‚free mode code', jsou skripty (šablony pro predikci a detekci neobvyklých stavů dat od ENERGOCENTRUM Plus) v textovém souboru txt, které lze nahrát či uložit ve Free modu.

14.3 Zdrojové kódy

Aplikace celkem využívá 5 skriptů.

Seznam skriptů:

1. Aplikace
    1. main.py – logika aplikace
    2. gui.kv – vzhled grafického rozhraní – propojení s main.py
2. Přídavné moduly
    1. neural_network.py – modul, ve kterém jsem objektově naprogramoval neuronové sítě MLP, HONU a učící algoritmy Backpropagation
    2. neural_network_functions.py – algoritmy predikce neuronové sítě využívající modul neural_networks.py. Tento skript není použit v diplomové práci (DP). Nicméně jsou v něm algoritmy, které jsem využil v DP a jelikož není součástí programu, nejsou uvnitř algoritmu nadbytečné řádky kódu, které komunikují s grafickým rozhraním. Na konci skriptu neural_network_functions.py jsou algoritmy implementovány na jednoduchých příkladech (spustí se odkomentováním jedné z funkcí example_x())
3. Dodatečné skripty
    1. learn_entr.py – skript, který se spustí při Run v modulu Learning error a který zobrazí data (yr.txt, yn.txt. e.txt, EAP.txt, EA.txt) ve složce evaluation

2. error_plot.py – **skript, který se spustí při Run** v modulu Output error a který zobrazí data (yr.txt, yn.txt. e.txt, MAE.txt, RMSE.txt) ve složce **evaluation**

Okomentování kódu

- Skripty neural_network.py a neural_network_functions.py jsou okomentovány důkladně, neboť na nich závisí implementace neuronových sítí
- Skripty main.py a gui.kv okomentuji pouze do té míry, aby bylo jasné chování programu
- Aplikace včetně okomentovaných skriptů bude též nahrána na github (https://github.com/marfay) **bez** dat ENERGOCENTRUM Plus