ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

FAKULTA
STROJNÍ

BAKALÁŘSKÁ PRÁCE

SUPERVIZOROVANÉ ALGORITMY STROJOVÉHO UČENÍ PRO ANALÝZU PRŮMYSLOVÝCH DAT

2019

ONDŘEJ
BUDÍK

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Budík** Jméno: **Ondřej** Osobní číslo: **458430**

Fakulta/ústav: **Fakulta strojní**

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

Studijní program: **Strojírenství**

Studijní obor: **Informační a automatizační technika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Supervizorované algoritmy strojového učení pro analýzu průmyslových dat

Název bakalářské práce anglicky:

Supervised machine learning algorithms for industrial data anlysis

Pokyny pro vypracování:

1. Proveďte rešerši úloh analýzy průmyslových dat (prediktivní údržba strojů, linek, detekce poruchových stavů, klasifikace stavů obecně, tj. co se dnes tím vlastně vše řeší...)
2. Proveďte rešerši publikovaných metod strojového učení používaného pro analýzu průmyslových dat (např. detekce por. Stavů, predikce veličin, por. Stavů,...,)
3. Na simulovaných datech (a případně dodaných reálných) otestujte a porovnejte vybrané metody (LNU, HONU, ELM, LSTM, s použitím knihoven nebo Vámi odvozených implementací).

Minimální rozsah 40 stran + přílohy.

Seznam doporučené literatury:

[1] PALADE, Vasile, Cosmin Danut BOCANIALA a L. C. JAIN, ed. Computational intelligence in fault diagnosis. London: Springer, 2006. Advanced information and knowledge processing. ISBN 978-1-84628-343-7.
[2] BUKOVSKY, Ivo a Noriyasu HOMMA. An Approach to Stable Gradient-Descent Adaptation of Higher Order Neural Units. IEEE Transactions on Neural Networks and Learning Systems [online]. 2016, 1–13. ISSN 2162-237X, 2162-2388. doi:10.1109/TNNLS.2016.2572310
[3] YANG, C., W. SHEN a X. WANG. The Internet of Things in Manufacturing: Key Issues and Potential Applications. IEEE Systems, Man, and Cybernetics Magazine [online]. 2018, 4(1), 6–15. ISSN 2333-942X. doi:10.1109/MSMC.2017.2702391

Jméno a pracoviště vedoucí(ho) bakalářské práce:

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

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

Datum zadání bakalářské práce: **26.04.2019** Termín odevzdání bakalářské práce: **12.06.2019**

Platnost zadání bakalářské práce: _____

doc. Ing. Ivo Bukovský, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

26. 4. 2019
Datum převzetí zadání

Podpis studenta

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

V Praze, dne _____ *Podpis* _____

Poděkování

Chci poděkovat vedoucímu této práce doc. Ing. Ivu Bukovskému, Ph. D. za jeho čas, který mi věnoval a také za pomoc, kterou mi poskytl při vypracování této práce a za vše, co jsem se díky němu naučil. Dále děkuji společnostem CompoTech PLUS s.r.o. a Howden ČKD Compressors s.r.o. za poskytnutí dat použitých v této práci, a děkuji i firmě mySCADA s.r.o. za otevřenost a konzultace problematiky průmyslových dat v počáteční fázi přípravy mé práce.

Velmi děkuji své rodině a nakonec i přátelům, kteří mě po celou dobu studia podporovali, kteří mi vždy pomohli jak mohli a kterým vděčím za veškeré své možnosti. Bez nich by to zcela jistě nebylo možné.

Souhrn: Cílem a výstupem této práce je rešerše aktuálních úloh analýzy průmyslových dat a otestování čtyř vybraných metod supervizorovaných neuronových architektur pro aplikace na obvyklé úlohy během analýzy průmyslových dat jako je porovnání jejich schopností predikce časových řad, vytvoření modelu systému, klasifikace a detekce neobvyklých stavů. Práce se zabývá různými přístupy jednotlivých neuronových sítí vzhledem k datům, na které jsou aplikovány. V závěru práce jsou shrnuty možnosti pro vhodné aplikace jednotlivých neuronových architektur na průmyslová data.

Klíčová slova: Neuronová síť, predikce časových řad, učící algoritmy, klasifikace, neobvyklé stavy, LNU,HONU, ELM, LSTM

Summary: The aim and the output of this work is a review of current problems of analysis of industrial data and testing of four selected methods of supervised neural architectures for applications on common tasks during analysis of industrial data such as comparison of their time series prediction capabilities, creation of the system model, classification and detection of unusual states. The thesis deals with different approaches of individual neural networks with respect to the data on which they are applied. The conclusion summarizes the possibilities for suitable applications of individual neural architectures to industrial data.

Key words: Neural networks, time series prediction, learning algorithms, unusual states, LNU, HONU, ELM, LSTM

Obsah

Úvod

Tématem zpracovaným v této bakalářské práci je rešerše aktuálních úloh v analýze průmyslových dat a dále jsou zde zpracovány supervizorované algoritmy strojového učení v aplikaci pro analýzu průmyslových dat. Konkrétně jde o algoritmy metod LNU, HONU, ELM a LSTM. Dané algoritmy jsou napsány a aplikovány v programovacím jazyce Python.

Jelikož jsou monitorovací systémy, např. Typu SCADA, již déle integrovány do výrobních linek velkého množství výrobců, kteří již déle umožňují monitorování a ukládání dat z jejich výrobních strojů, objevují se zde nové možnosti pro využití strojového učení v průmyslu pro oblasti predikcí a klasifikací poruch nebo opotřebení, které mají za cíl výrazně snížit náklady, které vznikají během případné neplánované odstávky linky, ale také mohou usnadnit optimalizaci výroby jako je například na datech založený návrh na častější servis dané části linky, jelikož algoritmus rozpozná, že zpomaluje zbytek výroby.

V teoretické části jsou uvedeny některé obecné poznatky o úlohách analýzy průmyslových dat. Je zde ve zkrácené formě popsána norma ISO 13374, která zajišťuje kompatibilitu a nezávislost na určitém výrobci v oblasti softwarových řešení pro analýzu průmyslových dat. Dále je zde popsáno rozdělení druhů prognostik používaných v průmyslu a problematika detekce poruchových stavů s jejími jednotlivými fázemi. Součástí této kapitoly je rovněž obecný popis problematiky detekce a klasifikace poruch, které musí být algoritmus schopný zpracovat. V druhé polovině této části jsou popsány především metody výpočetní inteligence, ale pro úplnost jsou zde zmíněny i principy a přístupy umělé inteligence, které se používají v analýze průmyslových dat. Zde jsou také všechny tyto metody charakterizovány a vysvětleny jejich funkční principy. Největší část problematiky výpočetní inteligence je věnována problematice neuronových sítí a jejímu aktuálnímu nasazení v průmyslové datové analytice, avšak pro úplnost obsahu výpočetní inteligence jsou zmíněny i metody fuzzy logiky a genetických algoritmů.

Dále jsou zde detailněji vysvětleny přístupy jednotlivých neuronových jednotek a jejich sítí. Je zde charakterizován neuron a jeho sítě a jsou zde také uvedeny základní metodiky trénování neuronových sítí. Poté detailněji popisuji jednotky a jejich sítě, které používám v dalších kapitolách. Jedná se o přístupy LNU, HONU, ELM a LSTM. V závěru teoretické části uvádím také mou volbu vývojového prostředí a jazyku.

V praktické části nejdříve popisuji data použitá v této práci a zabývám se jejich prvotním zpracováním, které obvykle spočívá v převedení na jiný, výhodnější, datový formát.

V případě dat společnosti Howden ČKD Kompresory, spol. s.r.o. jde o převod z velkého souboru Excel na soubory typu CSV a doplnění chybějících hodnot, které byly v rámci datové optimalizace vynechány. V případě předzpracování dat z vibrometru Polytec, jde o data zapůjčená firmou CompoTech PLUS, spol. s.r.o. bylo nutné napsání vlastní funkce k přístupu do speciálních binárních souborů, které tento vibrometr využívá, aby se tato data dala přečíst a bylo možné s nimi dále pracovat.

Popsané neuronové sítě jsem aplikoval na dvě obvyklé úlohy v datové analytice. Jako první jsem si zvolil úlohu predikce časových řad. Nejprve jsem otestoval všechny použité neuronové sítě nastavené k predikci na umělých datech, kde jsem ověřil jejich funkčnost a pomocí šumu jsem otestoval, zda se nejedná pouze o sledování průběhu. Poté jsem tyto sítě aplikoval na reálná data z kompresorů firmy HCKD. Jejich nastavení jsem ovšem ponechal velmi podobné zcela záměrně, aby bylo možné otestovat jejich schopnosti v reálném světě za obdobných nastavení a tím se ověřila komplexnost všech aplikací. K účelům ověření schopností klasifikace neuronových sítí jsem použil data z vibrometru a klasifikoval jsem 3 vybrané druhy kompozitních struktur s dostatečnou úspěšností. Závěrem každé aplikace jsem shrnul náročnost jejich aplikace, výhody a nevýhody a jejich úspěšnost.

1 Současný stav problematiky analýzy průmyslových dat

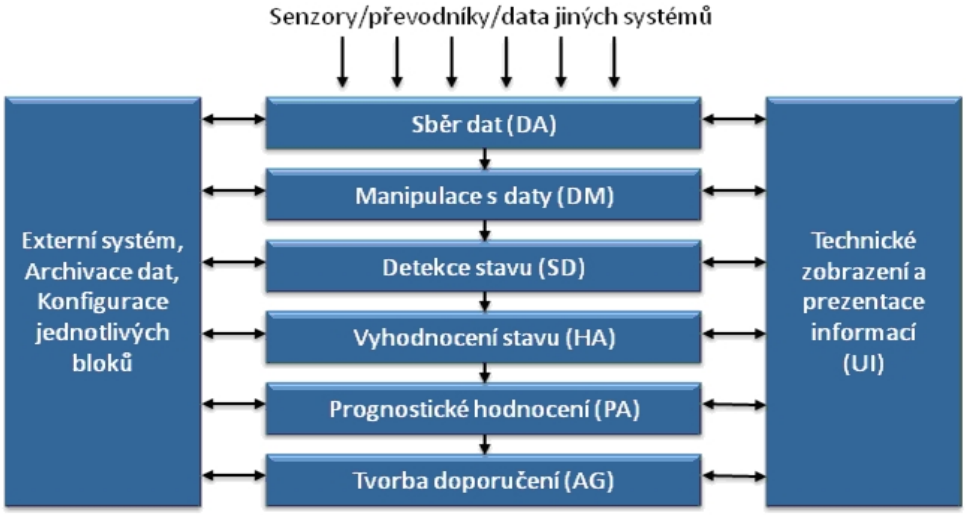
Jednou z hlavních úloh analýzy průmyslových dat je prediktivní údržba (Predictive Maintenance, PdM). Jde o druh údržby strojů, linek a ostatního průmyslového vybavení, který je založen na zpracování dat z jejich senzorů ve vztahu k jejich aktuálnímu nastavení. Tento přístup k údržbě slibuje úsporu nákladů oproti preventivním či korekčním údržbám. Jelikož je daný stroj či linka neustále monitorován a diagnostikován v reálném čase, lze údržbu naplánovat až v závislosti na stavu zařízení a tím snížit náklady nadbytečnou údržbou. Tato metoda má uvažovat degradaci částí strojů v závislosti na hodnotách ze senzorů a lze jí použít pro vyhodnocení aktuálního stavu od optimálního. Touto metodou lze zjistit, že například závěrové klapky stroje se během týdne zpomalí o tolik, že firma ušetří peníze jejich častější údržbou [1, 2].

V tomto odvětví zatím neexistuje žádný univerzální systém, který by byl schopný analyzovat a predikovat opotřebení pro jakoukoliv linku či výrobní stroj a jelikož je nutné vždy znát velké množství souvislostí, je tato práce jen obtížně plně automatizovatelná. Veškeré souvislosti musí datový analytik s pomocí expertů z výroby dát do určitého modelu závislostí a na tyto na sobě závislé bloky dále aplikovat nejvhodnější metody výpočetní či umělé inteligence. Bývá také obvyklé, že již pouhá vizualizace měřených dat dokáže odhalit velké množství nečekaných informací. Přesto velké množství společností stále nemá zavedenou vizualizaci dat.

Dalším problémem v současném průmyslu je velká rozmanitost ve formátech a způsobech uložení výrobních dat, což je před jejich zpracováním dalším velkým úkolem při vytěžování znalostí. Obvyklými problémy jsou unikátnost řešení sběru a ukládání dat v každé výrobě. Tento problém se snaží řešit norma ISO 13374, jejíž cíle jsou uvedeny v 1.1. Dalším problémem bývají nesprávně ukládané hodnoty nebo vypadávající senzory. Tyto dva problémy bývají zpětně obtížně řešitelné a v krajních případech mohou být i důvodem k celkovému znehodnocení dosavadního záznamu dat.

1.1 Norma ISO 13374

Cílem této normy je zajistit kompatibilitu různých komponent či jejich bloků na softwarové úrovni, tato norma poskytuje základní požadavky na software k umožnění bezproblémové komunikaci softwarových řešení od různých výrobců. Jejím cílem je platformě nezávislé předávání měřených dat či jiných informací z monitorování strojů, včetně jejich následného zpracování. Norma vešla v platnost roku 2012, bohužel se ovšem jedná o normu doporučenou, nikoliv závaznou. Na obrázku 1 je graficky zobrazeno propojení bloků systémů prediktivní údržby dle normy.



Obrázek 1: Zpracování informací a tok dat podle normy ISO – 13374; převzato z [1].

Základem každého systému analýzy dat dle normy jsou bloky funkcí DA (Data Acquisition), DM (Data Manipulation), SD (State Detection), HA (Health Assessment), které se starají o technickou diagnostiku a blok PA (Prognostic Assessment) poskytující prognostiku. Norma počítá i s nasazením bloku AG (Advisory Generation), který se obvykle spoléhá na expertní systém, či Neuro-Fuzzy, což je spojení fuzzy logiky a strojového učení. Dále norma specifikuje spojení jednotlivých bloků a grafického uživatelského rozhraní (User interface, UI) a také navázání na externí archivační či jiné systémy. Bezproblémová komunikace mezi těmito bloky má zlepšit bezpečnost, ulehčit údržbu, plánování provozu a minimalizaci odstavení zařízení bez závislosti na jednom výrobním systému [1].

1.2 Diagnostika a prognostika

Spojením diagnostiky a prognostiky můžeme s danou pravděpodobností odhadnout zbývající životnost (Remaining Useful Life, RUL) a dobu k poruše (Time to Failure, TF) na základě degradace jednotlivých komponent, nastavení provozu atp. [3]. Tento přístup vyžaduje rozšíření systému o informace, které se týkají degradace jednotlivých dílů a jejich význam pro chod systému jako celku. Nejde tedy pouze o selhání jednotlivých komponent, ale i o vliv jejich degradace či selhání na celý systém jako takový.

Prognostické metody lze na základě jejich principu rozdělit do třech základních skupin následujícím způsobem:

- Prognostika řízená daty (Data-Driven prognosis, DD)
- Prognostika založená na modelu stroje či linky (Model-Based prognosis, MB)
- Prognostika založená na statistických metodách (Probability-Based prognosis, PB)

Každá z těchto metod má své výhody a nevýhody, proto se v průmyslu nejčastěji využívá jejich kombinace.

1.2.1 Prognostika řízená daty

Tato úloha využívá a zpracovává data získané z monitorování daného zařízení jako jsou např: vibrace, teplota, tlak, napětí, proud, apod. Tato prognostika vychází z předpokladu, že charakteristika dat je dostatečně neměnná do té doby, než se v systému objeví selhání. Je tedy ze značné části založena na teorii rozpoznávání vzoru (Pattern recognition), ve které je k dispozici široké množství aparátů na zpracování dat. Mezi významnější metody aktuálně užívané v průmyslu patří:

- Analýza hlavních komponent (Principal Component Analysis, PCA), jež je vhodná pro redukci rozměru vstupních vektorů s co nejmenší ztrátou informace, která probíhá pomocí posuzování korelace mezi daty a jejich převedením na lineárně nekorelované hodnoty tzv. hlavní komponenty.
- Metoda nejmenších čtverců (Partial Least Squares regression, PLS regression), která je běžná pro lineární regresní analýzu.
- Kanonická variační analýza (Canonical-Correlation analysis, CCA), pracující na principu metody vícenásobné lineární regrese a korelační analýzy.

- Metody založené na neuronových sítích jako jsou například: pravděpodobnostní neuronové sítě (PNN, Probability Neural Networks), Lineární neuronové sítě (LNN, Linear Neural Networks), Neuronové sítě vyšších stupňů (HONN, Higher Order Neural Networks), Extreme learning machine ELM a Neuronové sítě s konvoluční vrstvou. Tyto metody jsou pro průmysl stále velmi nové a jejich nástup je pomalý, jelikož ve výrobě je velice důležitá stabilita, spolehlivost a jednoduchost aplikace. To prozatím není u neuronových sítí vždy jednoznačné z důvodů komplikované zpětné analýzy jejich neuronů a obvykle je nutná přítomnost dostatečně proškolené obsluhy.

Výhodou prognostiky řízené daty je možnost zredukovat velké množství informací do užitečných dat pro diagnostiku daného systému.

Nevýhodou je vysoká závislost na kvantitě a kvalitě získaných dat, která výrazně ovlivňuje efektivitu celé prognostiky.

Tento přístup je tedy velmi výhodný u systémů, jejichž chování nelze popsat modelem, anebo nelze nalézt dostatečně přesný matematický model [3].

1.2.2 Prognostika založená na modelu

Předpokladem pro tuto úlohu je, že na rozdíl od 1.2.1 je k dispozici velmi přesný matematicko-fyzikální model celého zařízení. Tyto metody využívají rezidua jako základní parametr pro odhad poruchy zařízení a také pro odhad jeho životnosti. Porovnáním výstupů z měřeného zařízení s jeho matematickým modelem získáme rezidua tohoto systému. Předpokladem této prognostiky je, že rezidua jsou značně velká v případě selhání a malá za normálního režimu (vzhledem k možným nepřesnostem a šumu v signálu). Velkou výhodou této prognostické metody je možnost využití fyzikálních znalostí systému do monitorování. Což umožňuje snížit počet monitorovaných veličin na minimální množství a další hodnoty odvozovat přímo z matematicko-fyzikálního modelu. Další výhodou je možnost implementovat do modelu i postupnou degradaci nástrojů či částí strojů na základě údajů ze senzorů, čímž se zvyšuje celková přesnost odhadu životnosti nebo náhlé chyby. Dobrá implementace této prognostiky je v oblasti na modelu založených návrhů strojů nebo výrobních linek, jelikož jsou často dostupné přesné matematické modely a tím pádem je k nim možné přidat i model degradace jednotlivých komponent [4].

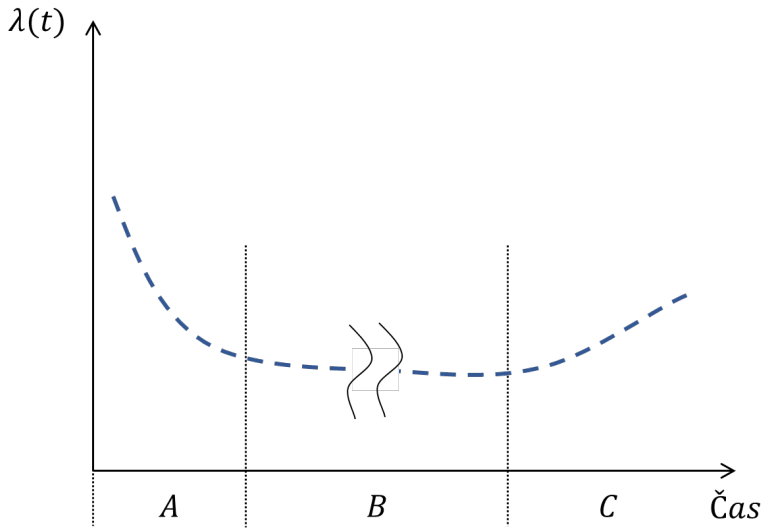
V této oblasti je aktuálně populární princip digitálního dvojčete, ve kterém je celá reálná linka propojena s její digitální replikou. Propojení je možné díky velkému množství senzorů,

které nemonitorují pouze stavy strojů linky během výroby, ale také ostatní stavy, jako jsou například okolní prostředí, nastavení linky, charakteristiky strojů v závislosti na prováděných úlohách a podobně. Všechna tato data jsou neustále posílána a zpracovávána digitálním dvojčetem. Ačkoliv díky postupně nastupujícímu internetu věcí (Internet of Things, IoT) pořizovací ceny takovéhoto množství senzorů s adekvátní síťovou strukturou se stále snižují, jedná se o stále velmi drahý aparát s omezenou podporou ze strany výrobců průmyslových zařízení [5].

1.2.3 Prognostika založená na pravděpodobnosti

Tato metodika má nejdelší historii z výše jmenovaných a je stále nejpoužívanější prognostickou metodou v průmyslu. Tato metoda si postačí i s málo detailními daty a spoléhá se především na distribuční funkce pravděpodobnosti (Probability Distribution Functions, PDF), které byly odvozeny na základě provozních údajů, statistických dat z historie a na základě výrobních parametrů.

Pro tuto prognostiku je typická takzvaná „vanová křivka“, která udává četnost poruch ve třech úsecích (graficky na obrázku: 2). Tento diagram představuje ukazatel intenzity poruch λ v závislosti na čase. Úsek A je úsekem častých poruch, tzv. období počátečního výpadku. Tyto poruchy jsou obvykle způsobeny nedostatky konstrukce, nedokonalosti ve výrobě nebo nevhodnost používání. Úsek B je úsekem ustáleného množství poruch, obvykle se jedná o náhodné příčiny nebo nedodržování provozních podmínek. Úsek C je úsekem dožití a únavy kam spadají únava, stárnutí či opotřebení [6].



Obrázek 2: Vanová křivka - představa o intenzitě poruch v čase, převzato a upraveno dle [6]. (Lambda představuje četnost poruch v čase)

1.3 Detekce poruchových stavů

Detekci poruchových stavů lze rozdělit na tři po sobě jdoucí fáze, a to na vlastní detekce poruchy (faulty detection), izolaci poruchy (faulty isolation) a analýzu poruchy (faulty analysis). Je důležité zmínit, že porucha je náhodný a jednoznačně neměřitelný proces, proto se můžeme o poruše dozvědět pouze na základě kombinace dat, které měříme na systému.

V první fázi, tedy detekci poruchy, je hlavním cílem zjištění času, kdy začala porucha působit. To je možné na základě znalosti běžné charakteristiky stroje s daným nastavením, anebo pomocí znalosti běžných hodnot okolních závislých senzorů. Mezi takové senzory patří například senzory vibrací, zvuků, teplot či infračervené kamery nebo jiné multispektrální senzory. Jejich vhodnou kombinací můžeme poté do určité míry určit počátek poruchového stavu. Často se vychází z nějaké databáze známých poruchových stavů.

Ve fázi izolaci poruchy jde hlavně o klasifikaci dané poruchy, tento proces je často založen na statistickém rozhodování a na matematické studii poruchových stavů. Tyto poruchové průběhy se obvykle ukládají do databáze určené ke klasifikaci známých poruch, která může být použita k metodám strojového učení. V této práci simuluje onu klasifikaci poruch klasifikace tři druhů kompozitů, jelikož principy zůstávají obdobné.

A v poslední fázi, analýzy poruch, jde o určení typu, velikosti a zdroje poruchy. Tato fáze je obvykle tou nejobtížnější, neboť její komplikovanost se velmi zvyšuje při vyšším počtu signálů v daném procesu, což v praxi také bývá limitující faktor.

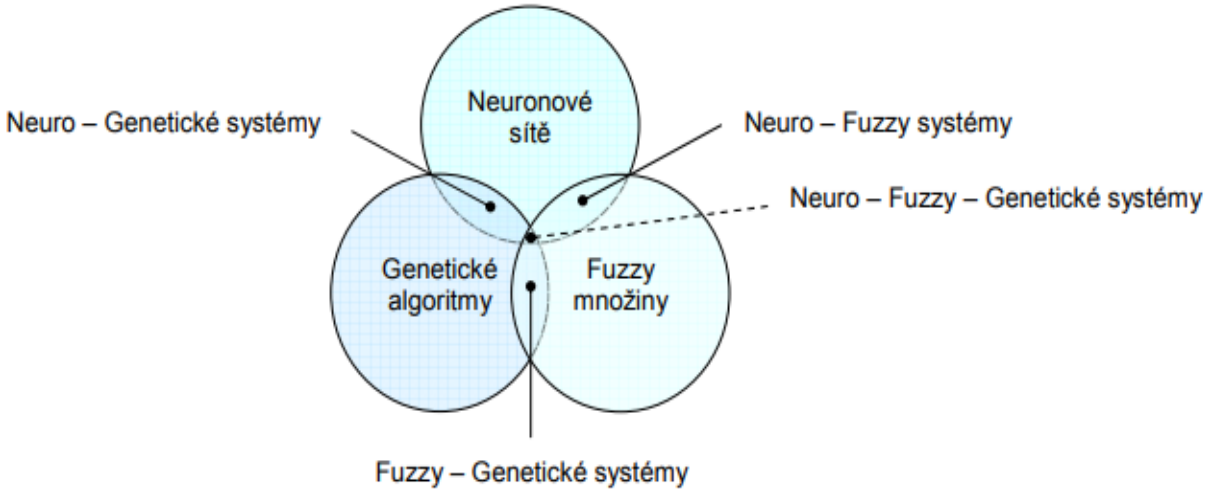
Proto se s nástupem strojového učení začínají prosazovat detekce založené na neuronových sítích, které jsou schopné zpracovávat velké množství dat velkou rychlostí. A jelikož je v průmyslu velmi důležitá stabilita a spolehlivost výsledků, jeví se supervizované algoritmy jako vhodnější nástroj k analýze, nežli jejich obtížně analyzovatelné nesupervizované protějšky. Přestože nesupervizované algoritmy mohou najít i závislosti, které nejsou z důvodů velkého množství senzorů zřejmé, je zde nejistota na základě čeho tak rozhodly a to je v průmyslu velmi velké riziko [7, 8].

1.4 Výpočetní a umělá inteligence pro analýzu průmyslových dat

Hlavním problémem, který je s analýzou průmyslových dat spojen je nelinearita dat, šum v signálech a ostatní nejistoty. Tyto problémy se dají řešit dvěma přístupy, a to

redundancí hardwaru, což znamená duplikaci senzorů nebo jiných fyzických zařízení, anebo analytickou redundancí, která používá redundantní funkční vztahy mezi daty systému. Jelikož je hardwarová redundance značně nákladná, je používána výhradně u velmi kritických systémů, kde může dojít k ohrožení života v případě jejich poruchy. Jde o zařízení jako jsou zařízení jaderných elektráren, turbo kompresory nebo letecký průmysl. V běžném průmyslu se tedy obvykle prosazuje redundantní datová analytika jako jsou přístupy výpočetní a umělé inteligence.

Přístupy výpočetní inteligence k např. rozhodování a řízení jsou charakterizované nesymbolickou reprezentací znalostí, která představuje vyjádření a práci s reálnými čísly nebo zpracování obrazových a zvukových informací. Do výpočetní inteligence patří fuzzy množiny, neuronové sítě, genetické algoritmy a jejich mutace. Na obrázku 3 jsou tyto oblasti graficky zobrazeny.



Obrázek 3: Oblasti výpočetní inteligence, převzato z [9] .

Přístupy umělé inteligence se snaží vytvářet stroje nebo systémy, které budou při řešení problému postupovat podobně jako člověk. Umělá inteligence jako taková se zabývá problematikou zpracování postupů a poznatků a jejich aplikaci na nový problém. Umělá inteligence je charakterizována symbolickou reprezentací znalostí a sekvenčním zpracováním. Symbolická reprezentace představuje jednoduché vyjádření v řeči symbolů a matematických zákonů. Aktuální oblasti použití jsou velmi široké a nemají jednotící metodu [3, 9].

1.4.1 Fuzzy množiny

Základní myšlenkou fuzzy množin je, že pokud nejsme schopni stanovit přesné hranice třídy vymezené vágním pojmem, nahradíme toto rozhodnutí mírou vybíranou z nějaké škály možností [10]. V průmyslu je fuzzy logika používána jak pro detekci pomocí modelování, tak izolaci poruch, pomocí klasifikace nelineárního systému. Obvykle jsou nasazeny fuzzy pravidlové systémy Mamdaniho typu, které jsou velmi oblíbené pro svou přehlednost díky lingvistické aproximaci. Téma fuzzy logiky je velmi dobře zpracované například v [3] nebo [11] a není tématem mé práce, avšak pro jeho důležitost bylo nutné ho alespoň krátce zmínit.

1.4.2 Neuronové sítě

Neuronové sítě reprezentují systémy zpracování informací, které se skládají ze vzájemně propojených jednoduchých pracovních jednotek, takzvaných neuronů. Každý neuron je nezávislá pracovní jednotka, která transformuje svůj vstup pomocí takzvané aktivační funkce. Propojení mezi jednotlivými neurony je charakterizováno pomocí hodnot vah, které představují paměť celé sítě. Tato problematika je dále popsána v kapitole 2.1. Mezi tři hlavní charakteristiky neuronových sítí, které je činní vhodnými pro modelování chování strojů či systémů, patří: schopnost generalizace, tolerance šumu a velmi rychlá odpověď po natrénování. Obecně, vektory vstupů a výstupů (input-output vectors) systému reprezentují senzory naměřené hodnoty a vždy obsahují určité množství šumu. Pokud tedy obsahují i trénovací data šum, neuronové sítě jsou stále schopné generalizace chování systému s úrovní přesnosti proporcionální k úrovni šumu. V průmyslu se neuronové sítě používají k detekci, izolaci a klasifikaci poruchy. Obvyklým přístupem je rozdělení celého systému na jeho menší závislé bloky, na které je poté aplikována neuronová síť nebo sítě. Nové a často používané neuronové sítě v průmyslu jsou uvedeny v tabulce 1.

Tabulka 1: Nové a často používané neuronové sítě v průmyslu (včetně hlubokých sítí) [3].

Druh neuronové sítě	Druh učení	Vhodný pro modelování	Vhodný pro klasifikace
Vícevrstvý perceptron (Multilayer Perceptron Network, MPN)	Supervizované	Ano	Ano
Rekurentní neuronové sítě (Recurrent Neural Network, RNN)	Supervizované	Ano	Ne
Neuronové sítě založené na principu radiálních bází (Radial Basis Function Neural Network, RBF-NN)	Nesupervizované	Ne	Ano
Dynamické neuronové sítě (Dynamic Neural Networks, DNN)	Supervizované	Ano	Ne
Oboustranná neuronová síť (Counter Propagation Networks, CPN)	Nesupervizované	Ano	Ano
Sítě samo-organizačních map (Self-Organizing Maps, SOM)	Nesupervizované	Ne	Ano
Bayesovské Pravděpodobnostní neuronové sítě (Probabilistic Neural Networks, PNN)	Supervizované	Ne	Ano

Neuronové sítě lze obecně rozdělit do dvou hlavních skupin podle jejich struktury, a to na dopředné šíření signálu a sítě se zpětnou vazbou.

V současnosti se v průmyslu využívají především struktury s dopředným šířením signálu, kde výstupy z jedné vrstvy jsou vedeny na vstup té další. Výstupem poslední vrstvy je tedy výstup této celé sítě.

Struktura sítí se zpětnou vazbou je odlišná v tom, že výstupy z vrstvy jsou vedeny zpět na vstup dané vrstvy. Tímto způsobem je možné realizovat výpočty založené na iteračním procesu a řešit tak i optimalizační úlohy [3, 12].

Mezi další neuronové architektury, které jsou často využívány patří:

- Lineární neuronové sítě (Linear Neural Networks, LNN) a jejich lineární neuronové jednotky (Linear Neural Units, LNU). Jejich hlavní výhodou je jejich jednoduchost a rychlost natrénování. Tyto sítě mohou poskytnout základní informace o chování zkoumaného systému a tím pomoci s rozhodnutím jakou další neuronovou síť aplikovat, pokud je požadována vyšší přesnost.
- Neuronové sítě vyšších stupňů (Higher Order Neural Networks, HONN) a jejich neuronové jednotky vyšších stupňů (Higher Order Neural Units, HONU). Tyto

jednotky dokáží velmi dobře modelovat i silně nelineární systémy a po natrénování jsou velmi rychlé. Jejich velkou výhodou je automatické zvolení počátečních vah pro nelineární data [13].

- Extrémní učící se stroje (Extreme Learning Machines, ELM). Tyto sítě mají využití v celé řadě odvětví od třídění dat přes rozpoznávání obrázků po předpověď časových řad. Tyto sítě jsou schopny se velice rychle naučit i velmi složité procesy za velmi krátký čas v porovnání s běžnými neuronovými sítěmi díky jejich modelu náhodné skryté vrstvy [14].

Sítě typu LSTM prozatím nejsou v průmyslu velmi využívány, jelikož jejich možné výhodné aplikace jsou teprve postupně objevovány. Jedná se o velmi komplikované celky, které působí obavy o stabilitu, a to dále zpomaluje jejich nasazení na výrobní data. Avšak po úspěších v oblastech jako jsou lingvistické zpracování, překladech nebo predikce cen akcií skepse k těmto komplikovaným celkům začíná upadat.

1.4.3 Genetické algoritmy

Genetický algoritmus je heuristický postup, který má za cíl pomocí aplikací principů evoluční biologie nalézt řešení pro obtížně analyzovatelné procesy. Tyto takzvané evoluční algoritmy se spoléhají na napodobování biologických evolučních procesů jako jsou dědičnost, mutace, přirozený výběr a křížení.

Principem genetického algoritmu je postupná tvorba generací, kde každý jedinec populace generace představuje jedno řešení a jejich kombinace je množina různých řešení daného problému. Na začátku simulace, v první generaci, bývá populace složena ze zcela náhodných členů, které jsou při přechodu do generace nové ohodnoceny takzvanou fitness funkcí, která vyjadřuje kvalitu řešení reprezentovanou tímto jedincem. Podle kvality určené touto funkcí jsou pak vybráni nejlepší jedinci, kteří jsou pak dále modifikováni evolučními procesy, čímž vznikne populace nová. Iterativním opakováním tohoto postupu se postupně zvyšuje kvalita řešení, dokud není dosažena postačující kvalita nebo není dosaženo předem nastaveného počtu generací či uplynulé doby [15, 16].

Genetické algoritmy se díky své povaze používají v průmyslu především k optimalizaci výrobních procesů. Příklady jejich aplikace pro tváření kovů, výroby papíru, plánování výroby, v chemickém průmyslu anebo optimalizace povrchu jsou uvedeny v [17]. V oblastech datové analytiky se tyto algoritmy neprosazují.

1.4.4 Přístupy umělé inteligence

Cílem umělé inteligence je napodobit lidskou mysl během uvažování nad problémem, což znamená na základě natrénovaných znalostí tyto znalosti aplikovat na nový, dříve neviděný problém. Často aplikovanou metodou je takzvané hluboké strojové učení, také známé jako hluboké neuronové sítě (Deep Neural Networks, DNN). Tyto sítě se mohou učit supervizovaně, semi-supervizovaně ale i nesupervizovaně, čímž jsou zajímavé k řešení velmi rozdílných problémů. Hlavním rozdílem oproti normálním neuronovým sítím je přítomnost vícerých skrytých vrstev což umožňuje natrénování i velmi komplexních problémů. Například během zpracování obrazu platí, že čím víc vrstev daná hluboká neuronová síť má, tím více prvků (Features) je schopná rozpoznat. Avšak s větším počtem roste exponenciálně i počet nutných dat k jejich natrénování [18].

V průmyslu mezi průkopníky v nasazení umělé inteligence do výrobních strojů patří firma OKUMA s jejich CNC zařízením OSP-P300A do kterého implementovali diagnostický modul s umělou inteligencí, který již obsahuje základní mechanické vlastnosti zařízení, CNC analytická data a data diagnostiky řízení. Jejich modul spolehlivě odhaluje lokální náhodné anomálie a dokáže i navrhnout vhodný čas k údržbě, aby se předešlo poškození zařízení [19].

2 Použité metody

V této práci se budu zabývat aplikací čtyř neuronových sítí na umělá, ale i poskytnutá reálná data z vibrometru Polytec od firmy CompoTech PLUS, spol. s.r.o. a data z turbokompresorů firmy Howden ČKD compressors, spol. s.r.o. Dále popsané sítě jsem porovnal na jejich schopnosti modelování daného systému z použitých dat a poté jsem otestoval jejich klasifikační schopnosti.

2.1 Definice neuronů a jejich sítí

Neuron je základní výpočetní jednotka neuronových sítí a je tvořen pomocí trojce $(f, \vec{w}, t)$ kde

- f je přenosová funkce $R \to R$
- $\vec{w} \in R^n$ je vektor vah

- $t \in R$ je práh

Neuron pro svůj vstup $\vec{x} \in R^n$ počítá výstup $y \in R$ kde horní index T označuje transpozici

$$y = f(wx^T + t) \tag{1}$$

Neuronové sítě vznikají propojováním neuronů mezi sebou, kde výstup jednoho neuronu může sloužit jako část vstupu neuronu druhého. Neuronovou sítí se rozumí dvojice (N, C) kde

- N je množina neuronů
- $C \subseteq N \times N$ je orientovaná množina spojů

Obecná neuronová síť obsahuje i směry vedoucí zpět na předchozí neuron, což ovšem většina v průmyslu užívaných sítí nevyužívá (kromě sítí založených na RNN) a jak již bylo řečeno v 1.4.2 užívané jsou prozatím především sítě dopředné. Tyto dopředné sítě jsou obvykle nazývány vrstevnaté neuronové sítě. Spoje těchto sítí vedou vždy z nižší vrstvy l_x do vrstvy vyšší l_{x+1} [20].

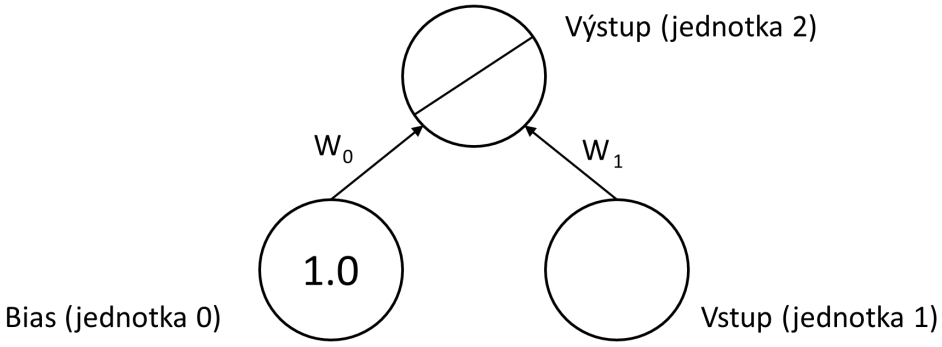
2.2 Lineární neuronové jednotky, LNUs

Lineární neuronové jednotky jsou jednoduchou aplikací neuronu nebo jejich sítí na data s cílem zjistit, či predikovat lineární vztahy v systému. Ačkoliv je v průmyslu lineárních závislostí omezený počet, vyplatí se vždy podívat, jak si s danou problematikou tato metoda poradí, jelikož tato architektura je velmi robustní a méně náchylná na chyby v datech, a je také dobré zjistil, zda nám její přesnost nepostačuje.

Lineární jednotky se obecně dají vyjádřit modelem na obrázku 4. Ten se skládá ze vstupní jednotky, která poskytuje externí hodnoty x celé síti, dále z jednotky bias, která zůstává po celou dobu konstantní, obvykle na hodnotě 1. Lineární výstupní jednotka poté vypočítá jejich součet:

$$y = x \cdot w_1 + x_0 \cdot w_0 \ ; \ kde \ x_0 = 1, \tag{2}$$

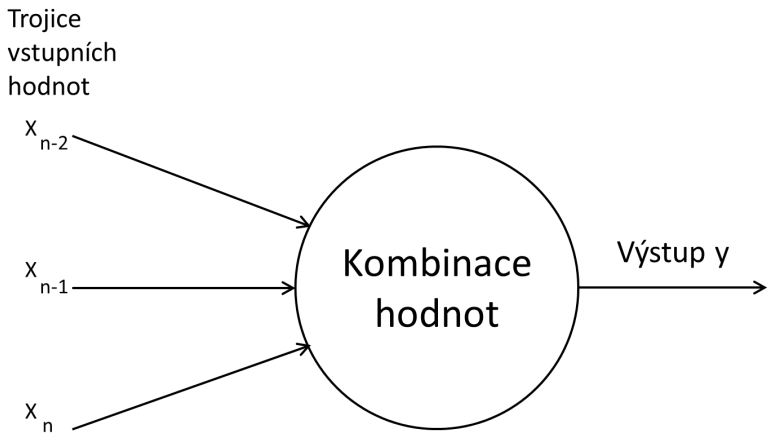
na této rovnici je dobře vidět, že jde opravdu pouze o ekvivalent rovnice (1).



Obrázek 4: Model lineárního neuronu.

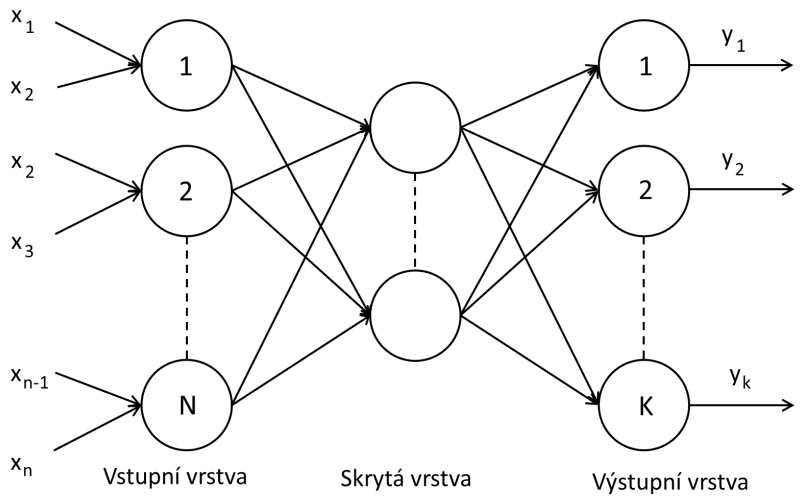
2.3 Neuronové jednotky vyšších stupňů, HONUs

Neuronové jednotky vyšších stupňů vycházejí z vrstevnatých neuronových sítí, což znamená, že mají pouze dopředné spoje. Oproti lineárním neuronovým sítím, které mají vstupní vrstvu stejné délky jako je vstupní vektor, je vstupní vektor pro neuronové sítě vyšších řádu rozdělen na *N* hodnot, které jsou předány do jednoho vstupního neuronu a ten tyto hodnoty spolu kombinuje. Počet vstupních neuronů odpovídá počtu všech *N*-tic ze vstupního vektoru. Na obrázku 5 je pro názornost graficky zobrazen neuron třetího řádu, který tyto vstupní hodnoty kombinuje. Jednou ze vstupních hodnot opět bývá bias. Možnou kombinační metodou vstupu je například vynásobení všech vstupních hodnot.



Obrázek 5: Vstupní neuron třetího řádu.

Neuronové sítě se liší oproti lineárním pouze vstupní vrstvou, proto na ně lze aplikovat stejné techniky a algoritmy pro učení. Diagram možné stavby neuronové sítě druhého stupně dle architektury HONU je k vidění na obrázku 6, a až na vstupní vrstvu je zcela stejný jako sítě LNU.



Obrázek 6: Příklad neuronové sítě vyšších řádu.

Dle [13] je klasická notace HONU forma $\sum\prod$, kde například HONU druhého stupně, což je QNU (Quadratic Neural Unit) je následující:

$$\tilde{y} = \sum_{i=0}^{n} \sum_{j=i}^{n} w_{i,j} \cdot x_i \cdot x_j, \tag{3}$$

kde $\tilde{y}$ je neurální výstup a rozšířitelný vstupní vektor je

$$\mathbf{x}(k) = [x_0 \;\; x_1 \;\; \ldots \;\; x_i \;\; x_{i+1} \;\; \ldots \;\; x_n]^T, \tag{4}$$

kde $x_0 = 1$ je bias (rozšiřující jednotka pro neurální práh), díky čemuž je možné HONU použít i pro polynomy nižších stupňů. Dále, jak je uvedeno v [21], lze dlouhá vektorová forma HONU zapsat pomocí $col^r(x)$ a $row^r(x)$. Pro dříve uvedenou QNU je aplikace následující

$$\tilde{y} = \mathbf{w} \cdot col^{r=2}(\mathbf{x}) = row^{r=2}(\mathbf{x}) \cdot \mathbf{w}^T, \tag{5}$$

kde operátory $col^r(\mathbf{x})$ a $row^r(\mathbf{x})$ transformují vstupní vektor $\mathbf{x}$ do dlouhého vektoru polynomiálních výrazů, což je do vektoru dlouhého sloupce nebo řady

$$col^{r=2}(\mathbf{x}) = \{x_i \cdot x_j \;;\; i = 0 \ldots n,\; j = i \ldots n\}, \tag{6}$$

obdobně jako je transformován vstupní vektor $\mathbf{x}$ jsou i neurální váhy vektorizovány do 1-D vektoru w což v případě QNU znamená následující

$$\mathbf{w} = [w_{0,0} \;\; w_{0,1} \;\; \ldots \;\; w_{i,j} \;\; \ldots \;\; w_{n,n}] = \begin{bmatrix} w_{i,j} & i = 0 \ldots n \\ & j = i \ldots n \end{bmatrix}, \tag{7}$$

obdobně se tyto dlouhé vektory neurálních vah používají i pro vyšší polynomiální stupně. Způsob učení vah HONU je velmi obdobný jako u LNU a obecně je uveden jako

$$\mathbf{x}(k) = \mathbf{w}(k-1) + \Delta\mathbf{w}, \tag{8}$$

kde k označuje index vzorku nebo index epochy pro dávkové učení.

Objektově naprogramované sítě HONU, tedy LNU a QNU jsou uvedeny v příloze 7.3 ve skriptu neural_units.py

2.4 Neuronové sítě s náhodnou skrytou vrstvou ELM

Tyto neuronové sítě řeší problém sklouznutí do lokálního minima, které je problematické u metody gradient descent během učení metodou batch a také řeší problematiku zdlouhavého učícího procesu. V případě ELM se volí váhy skrytých vrstev pouze na začátku učícího procesu zcela náhodně a poté zůstávají zcela neměnné. Jediné váhy, které ne nutné natrénovat, jsou ty výstupní, což je možné provést pomocí metod jednoduché lineární regrese [22]. Model neuronu pro ELM je

$$y_n = \mathbf{W_{out}} \cdot f(\mathbf{W_{in}}\mathbf{x}), \tag{9}$$

kde y_n je neurální výstup, f je aktivační funkce, w_{in} je vektor vstupních vah, x je vstupem do neuronové sítě a w_{out} je matice výstupních vah.

Jak již bylo řečeno, vstupní váhy, tedy w_{in} se zvolí náhodně a zůstanou konstantní. Váhy w_{out} se také zvolí náhodně ale budou se adaptovat. Určení výstupních vah spočívá v určení matice H, která reprezentuje výstup ze skryté vrstvy což je

$$\mathbf{H} = f(\mathbf{w_{in}}\mathbf{x}). \tag{10}$$

Poté se spočítá Moore-Penroseova pseudo-inverzní matice $\mathbf{H} \rightarrow \mathbf{H}^{\dagger}$ a vypočte se výstup z neuronové sítě pomocí vzorce

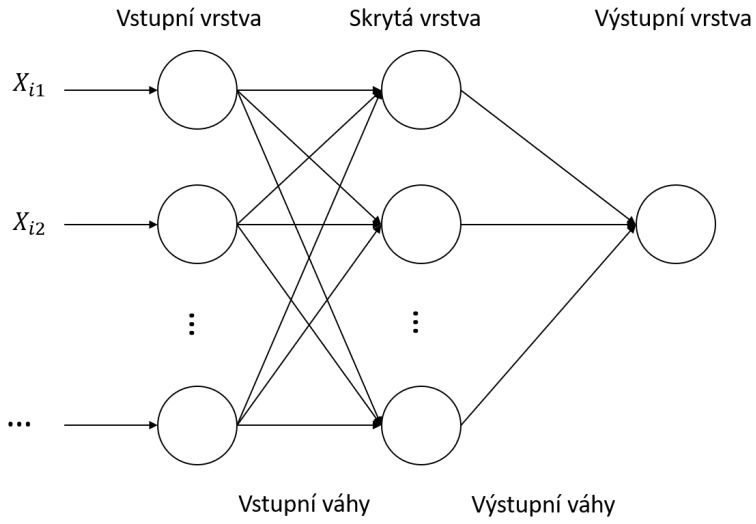
$$y_n = \mathbf{W_{out}}\mathbf{H} \tag{11}$$

a nakonec se přepočtou hodnoty matice vah w_{out} následovně

$$\mathbf{W_{out}} = \mathbf{H}^{\dagger} y_n. \tag{12}$$

Výpočetní obtížnost trénování ELM je dle autora [22] velmi malá a díky tomu je i čas nutný k jejímu natrénovaní velmi nízký s velice dobrými výsledky. Jako aktivační funkce je obvykle využíváno funkcí sigmoid nebo tanh. U ELM je tedy jediným nastavitelným

parametrem velikost skryté vrstvy a výstupní aktivační funkce. Toto nastavení má vliv na správnosti výsledku, jelikož příliš malá síť nemusí být schopná pojmout prudší změny vstupů, a naopak příliš velká síť má za následek delší učící čas, ale i nesprávný výsledek vlivem výchylky aproximace. K určení nejvhodnější velikosti skryté vrstvy lze do procesu vložit algoritmus, který postupně zvyšuje nebo snižuje velikost skryté vrstvy, jak je uvedeno například v [23]. Model sítě typu ELM je uveden na obrázku 7.

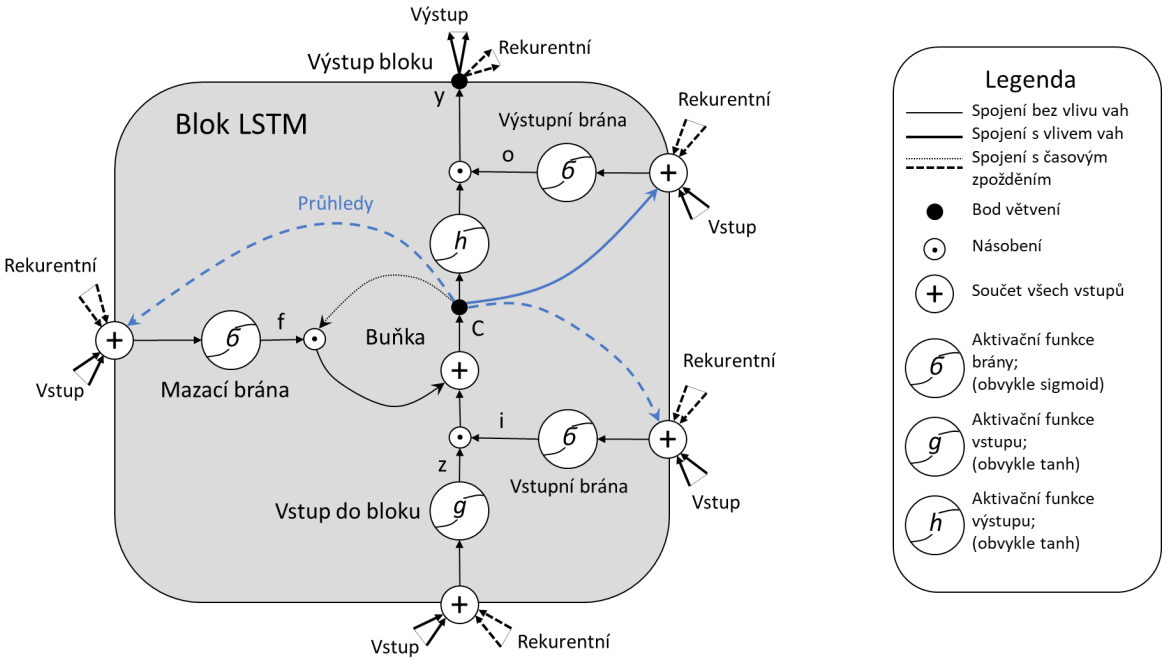


Obrázek 7: Model sítě typu ELM.

2.5 Long Short-Term Memory, LSTM

Long Short-Term Memory neboli Dlouho Krátko-Dobá Paměť je druhem rekurentních neuronových jednotek, které jsou stavebními jednotkami pro mnohem složitější celky. Běžné rekurentní sítě se spoléhají na metodu zpětného šíření tzv. Backpropagation, přesněji zpětné šíření skrz čas (backpropagation through time, BPTT), která slouží těmto neuronovým sítím jako krátkodobá paměť a její princip je vysvětlen v [24]. Tento přístup ovšem neuchovává důležité informace příliš dlouho, dle autora [25] jde obvykle o 10 – 12 kroků, kdežto architektura LSTM je schopná uchovávat informace z mnohem více kroků, a to přes 1000. LSTMs uchovávají informace mimo normální běh rekurentních sítí v uzavřené buňce za vstupní a výstupní bránou. Tato buňka funguje podobně jako počítačová paměť. Může do ní být zapisováno, čteno, mazáno, ale o tom, co bude zapsáno, kdy dovolí čtení a kdy vymazání rozhoduje buňka sama na základě bran které se otvírají nebo zavírají. Tyto brány jsou analogové, vybavené elementárním násobením sigmoid a jsou vždy v rozsahu od 0 do 1. Tyto

brány fungují jako další neuronové jednotky, které v závislosti na vstupu, výstup potlačí, anebo zesílí. Mají tedy své vlastní váhy, které jsou nastavovány během učícího procesu rekurentní sítě. Model na obrázku 8 graficky ukazuje, jak data prochází blokem LSTM.



Obrázek 8: Detail modelu bloku LSTM, překresleno podle [35] a upraveno.

2.5.1 Zpracování dat sítěmi s neurony typu LSTM

Prvním krokem v LSTM je obvykle rozhodnutí, které informace LSTM neuron zapomene. O to se stará funkce mazací brány, která je

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \tag{13}$$

kde σ je aktivační funkce brány, obvykle sigmoid, W_f jsou váhy této sítě, b_f je její bias, x_t je aktuální hodnota vstupu a h_{t-1} je předchozí stav buňky.

Poté následuje rozhodnutí, jaká nová data do buňky uložíme. Tento krok se skládá ze dvou částí, a to z části vstupní brány i_t, která je funkcí sigmoid a rozhoduje jaké hodnoty budeme aktualizovat a z části tanh, který tvoří vektor kandidátních hodnot $\widetilde{C}_t$, které mohou být přidány do aktuálního stavu. Hodnotu brány a vektor kandidátních hodnot se vypočítá následovně

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right), \tag{14}$$

$$\widetilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_c\right), \tag{15}$$

kde obdobně σ a $\tanh$ jsou aktivační funkce brány, W_i, W_C jsou jejich váhy, b_i, b_c jsou hodnoty bias, x_t je aktuální hodnota vstupu a h_{t-1} je předchozí stav buňky. Dále aktualizujeme starý stav buňky C_{t-1} do stavu aktuálního C_t pomocí

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t. \tag{16}$$

A na konec se LSTM musí rozhodnout, jaká data bude mít na svém výstupu. Tento výstup bude založený na filtrovaném výstupu stavu buňky C_t, filtrování provádí výstupní brána, která má obdobnou rovnici jako brány mazací a vstupní. Což je

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{17}$$

a funkce výstupní

$$h_t = o_t * \tanh\left(C_t\right). \tag{18}$$

Často používanou variací této obecné architektury je založena na článku autora [26], kde přidání tzv. průhledů (peepholes) do rovnic bran, jak je vidět modrými spojeními na obrázku 8, zvýší úspěšnost této RNN v rozeznávání sekvencí bez nutnosti alespoň krátkého trénovacího vzorku. Modifikace rovnic těchto bran vypadá tedy následovně

$$\begin{aligned} f_t &= \sigma\left(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f\right), \\ i_t &= \sigma\left(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i\right), \\ o_t &= \sigma\left(W_o \cdot [C_t, h_{t-1}, x_t] + b_o\right). \end{aligned} \tag{19}$$

Trénování sítě nebo jednoho bloku LSTM probíhá podobně jako u LNU nebo HONU. Avšak vzhledem ke komplexnosti systému, je výhodné celý tento proces optimalizovat. Možnosti optimalizací jsou popsány autorem v [27].

Implementace sítí ELM a LSTM jsou uvedeny v příloze 7.3 ve skriptu main.py v náležitě pojmenovaných a okomentovaných funkcích.

2.6 Vybrané metody učení neuronových sítí

2.6.1 Gradient descent

Jednou možností k natrénování lineární neuronové sítě je metoda klesání podle gradientu (Gradient Descent, GD). K tomu musíme být schopní vypočítat gradient G ztrátové funkce vzhledem ke každé váze w_{ij}. Tento gradient nám říká, jak malá změna právě té váhy nám ovlivní celkovou chybu E. Jde tedy o nalezení minima této chybové funkce. Výchozí rovnice pro výpočet gradientu jsou

$$E = \sum_p E^p, E^p = \frac{1}{2} \sum_i (y_t^p - y_n^p)^2 \tag{20}$$

kde p označuje aktuální trénovací bod, nikoliv umocnění a i je počet všech vah. Po jejich odvození pro neuronovou síť, které je uvedeno v [28], získáme rovnici pro adaptaci jednotlivých vah i kterou lze zapsat včetně chybové funkce takto

$$w_i\,(k+1) = w_i\,(k) - \frac{1}{2} \cdot \mu \cdot \frac{\partial e^2}{\partial w_i}. \tag{21}$$

K výpočtu gradientu G celé sítě, sečteme jednotlivé podíly každé dané váhy podle rovnice (21) přes všechna data. Poté můžeme odečíst malý podíl μ, také známou jako hodnota učení (learning rate) ze všech vah od G a tím provést metodu gradient descent a získat nové, lehce upravené váhy.

Kroky výpočtu tohoto krokového algoritmu jsou následující:

1. Inicializace
   spočívá v nastavení všech vah na malé náhodné hodnoty (doporučují se hodnoty mezi -0,05 a 0,05).
2. Opakování
   následujících kroků, dokud není dosaženo požadované přesnosti nebo přednastaveného počtu epoch trénování.
   1. Pro každou váhu w_{ij} se nastaví $\Delta w_{ij} := 0$.
   2. Pro každý bod v datech $(x, t)^p$
      1. nastavení vstupních jednotek na x
      2. vypočtení hodnot výstupních jednotek.

3. Pro každou váhu w_{ij} se nastaví $\Delta w_{ij} := \Delta w_{ij} + (t_i - y_i)y_j$.

3. Pro každou váhu w_{ij} se nastaví $w_{ij} := w_{ij} + \mu \Delta w_{ij}$.

Obvykle algoritmus končí, pokud jsme dosáhli $G = 0$ nebo jsme v jeho blízkém okolí, což znamená že algoritmus zkonvergoval, anebo pokud bylo dosaženo počtu kroků opakování.

2.6.2 Levenberg-Marquardt

Algoritmus LM byl vyvinut nezávisle na sobě Levenbergem a Marqurdtem. Tento algoritmus je iterační metoda, která hledá minimum chybové funkce. Metoda LM se stala standardní nelineární metodou nejmenších čtverců, která je kombinací metod Gradient Descent a Gauss-Newton. Ze začátku postupuje LM jako gradient descent, tj. pomalu ale s garancí konvergence k minimu, když je řešení blízko, stane se LM metodou Gauss-Newtonovou a konvergence k minimu se značně urychlí. Tato metoda patří do skupiny dávkového učení a váhy se aktualizují vždy přes celá data [29].

LM dosahuje vyšší rychlosti učení bez nutnosti výpočtů Hessovi matice. Hessova matice je čtvercová matice druhých parciálních derivací skalární funkce [30]. Autoři LM uvažují hledání minima chybové funkce jako sumy kvadrátu chyb,

$$E = \frac{1}{2} \sum_{N}^{k=0} e^2(k), \tag{22}$$

a díky této úvaze, lze pak aproximovat Hessovu matici jako

$$\mathbf{H} = \mathbf{J}^T \cdot \mathbf{J}, \tag{23}$$

kde gradient se spočte jakoukoliv

$$\mathbf{g} = \mathbf{J}^T \cdot \mathbf{e}, \tag{24}$$

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 biasu.

Algoritmus LM využívá aproximaci Hessovi matice pro adaptaci vah následovně

$$\mathbf{w} = \mathbf{w} - \left[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right]^{-1} \mathbf{J}^T\mathbf{e}, \tag{25}$$

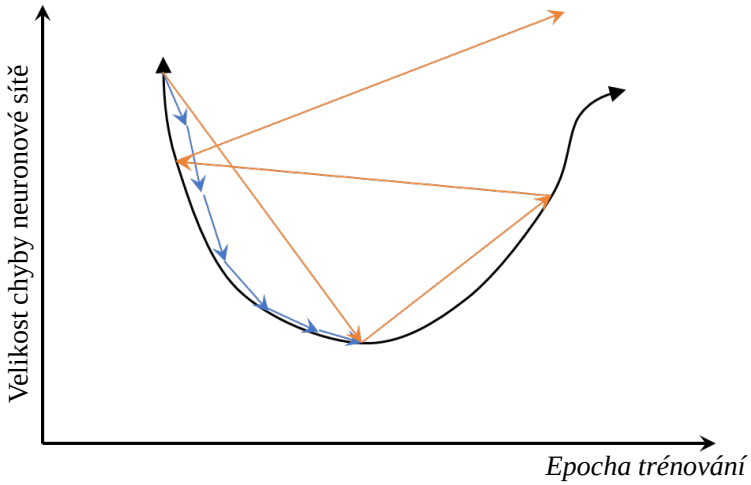
kde $\mathbf{I}$ je jednotková matice.

Pokud je tedy rychlost učení $\mu = 0$, pak se LM chová jako Newton-Gaussova metoda. Pokud je rychlost učení μ velké, pak se bude LM chovat jako Gradient Descent s malým krokem. Po každém iteračním kroku se hodnota μ zmenší a v určitém momentu začne převažovat Newton-Gaussova metoda [31].

Aplikovaný algoritmus GD a LM jsou uvedeny v příloze 7.3 ve skriptu neural_units.py

2.6.3 Rychlost učení a jeho metody

Během učení vah neuronové sítě je velmi důležité správné nastavení rychlosti učení (learning rate) μ, která určuje, jak moc velkou změnu po jedné epoše provedeme na váhách w, jak můžeme vidět v kroku 2.3. algoritmu gradient descent 2.6.1. Pokud bude μ příliš malé, bude trénování algoritmu trvat velmi dlouho, než nastane konvergence. Naopak pokud bude μ příliš velké, může nastat divergence naší neuronové sítě, což obvykle končí přetečením paměti počítače. Grafické znázornění nevhodné rychlosti učení je na obrázku 9.



Obrázek 9: Grafické zobrazení rychlosti učení.
Modře je uvedeno příliš malé μ.
Oranžově je uvedeno příliš velké μ.

K učení můžeme přistupovat pomocí dvou metod, a to pomocí

- Dávkového učení (batch learning), které spočívá v učení na celém souboru dat najednou, tedy výpočtu příspěvků gradientu a až po poslední vstupní hodnotě jsou váhy aktualizovány.
- Učení po lince (online learning), přičemž jsou váhy aktualizovány po každé vstupní hodnotě. Jelikož gradient pro jediný bod může být ovlivněn šumem, nazývá se tato metoda někdy také stochastický gradient descent.

Mezi hlavní výhody metody online learning patří:

- Podstatně vyšší rychlost natrénování, obzvláště na systémech s velkým množstvím dat.
- Lze použít i na nefixní počet dat (pokud data ze senzorů stále přibývají).
- Výhodnější pro modelování nestacionárních prostředí.
- Šum, který je součástí gradientu nám může pomoci uniknout z lokálního minima.

3 Volba vývojového prostředí

Vývojová prostředí, která by nejvíce vyhovovala naprogramování klasifikačního a predikčního nástroje a jsou vhodná pro náročné výpočty, jsou Python a Matlab. V obou případech je možné rychle tvořit nové funkce a možnosti obou prostředí jsou pro tento účel široké.

Pro zpracování mé bakalářské práce jsem si zvolil Python. Zdůvodnění je uvedeno v bodech níže:

- Neuronové sítě lze naprogramovat zcela objektově, tedy tak aby práce s nimi byla jednodušší a co nejvíce automatizovaná. Je zde také možnost použití širokého množství knihoven k usnadnění práce s neuronovými sítěmi. Dále také z pohledu objektového programování je Python vybavenější než Matlab.
- Python je na rozdíl od Matlabu open-source a vše co jsem v mé bakalářské práci vytvořil mohu dále používat i pro jiné projekty.
- Python se pro mne stal hlavním programovacím jazykem a využívám ho i pro své další projekty. Objektově naprogramované knihovny neuronových sítí budu tedy moci použít i v budoucnosti.
- Python má široké možnosti konektorů na další programy i v jiných jazycích. Budu tedy moci v budoucnu použít výpočetní kvality Pythonu i pro jiné programy v mých budoucích projektech.

Jako editor jazyku Python jsem si vybral program Spyder, jelikož je optimalizovaný pro kód aplikovaný na obtížné výpočty, a navíc je také nainstalován během instalace balíčku Anaconda.

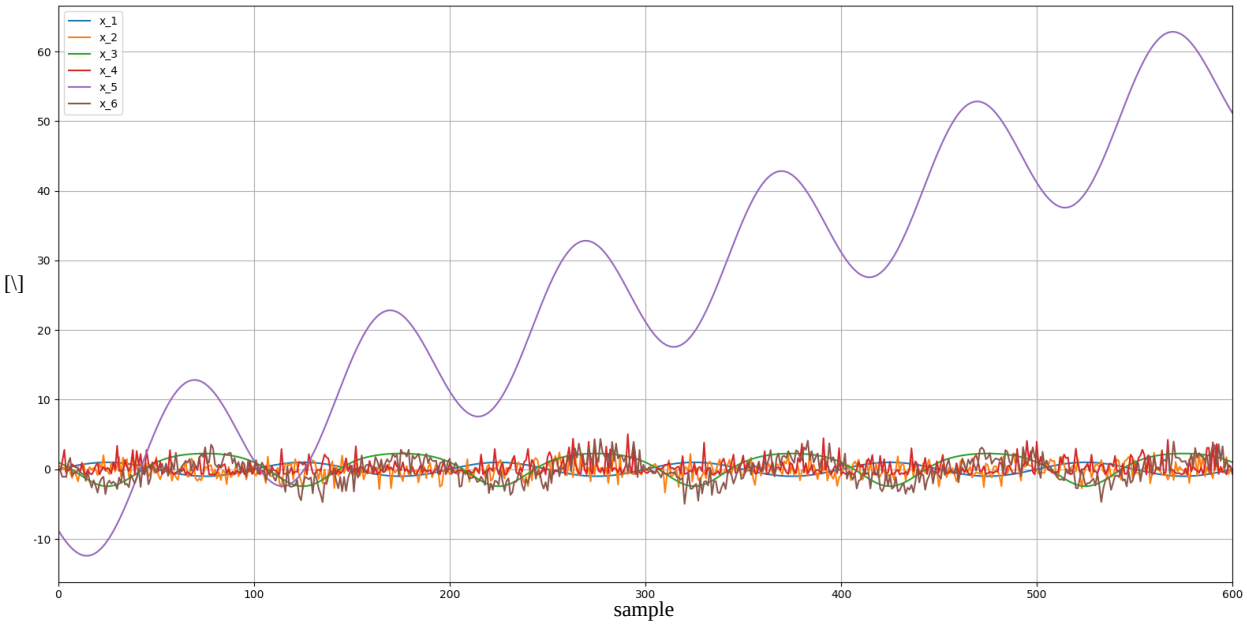
Instrukce k instalaci použitého prostředí jsou uvedeny v příloze 7.1.

4 Popis použitých dat

Data zpracovávána v podkapitolách 5.1, 5.2 jsou níže detailněji popsána. Vysvětluji jejich vznik nebo původ a jejich nutné základní předzpracování, aby s nimi bylo možné dále pracovat v metodách aplikací neuronových sítí. Prvotní předzpracování se obvykle skládá z kroků převodu dat různých formátů na jednotný formát, který je rychle přístupný obecnými metodami. Jedná se například o převod většího souboru dat v nevhodném formátu, jak je uvedeno v 4.3, do jiného. V mých aplikacích jsem nepracoval s daty formátu BigData, proto jsem si vystačil s formátem CSV a neaplikoval jsem více optimalizované formáty jako je například HDF5 jehož princip je blíže vysvětlen v [32].

4.1 Umělá data

Vytvořil jsem umělá dat, která mezi sebou mají vzájemné souvislosti a časové posuny mezi jejich průběhy. Tato data ověřují schopnost algoritmů vytvořit model systému a na jeho základě být schopni jeho predikce. Umělá data jsou zobrazena na obrázku 10.



Obrázek 10: Umělá data, časové průběhy 6 veličin se vzájemnými souvislostmi a časovými posuny.

Tyto veličiny jsou definovány následovně

$$\mathbf{X} = [\mathbf{x_1} \quad \mathbf{x_2} \quad \mathbf{x_3} \quad \mathbf{x_4} \quad \mathbf{x_5} \quad \mathbf{x_6}] \tag{26}$$

kde

$$x_1(k) = \sin\left(\frac{2 \cdot \pi \cdot \Delta t}{10} \cdot k\right), \tag{27}$$

$$\mathbf{x}_2 = randn(N), \tag{28}$$

$$x_3(k) = 3 - 2 \cdot e^{x_1(k)}, \tag{29}$$

$$x_4(k) = x_2(k-7) + \frac{x_2^2(k-7)}{3}, \tag{30}$$

$$x_5(k) = -x_1(k+8) \cdot 10 - 4 + \frac{k}{10}, \tag{31}$$

$$x_6(k) = x_2(k-10) - 2 \cdot x_1(k-5), \tag{32}$$

kde pro konstantní vzorkování $t \leftarrow \Delta t \cdot k; k = 1, 2, \ldots N$ (v Pythonu $k = 0, 1, \ldots N-1$) pro $\Delta t = 0.1$.

Skript použitý ke generování dat včetně dat použitých v této práci je uveden v 7.3 pod umela_data.py

4.2 Kompozitní struktury měřené vibrometrem Polytec

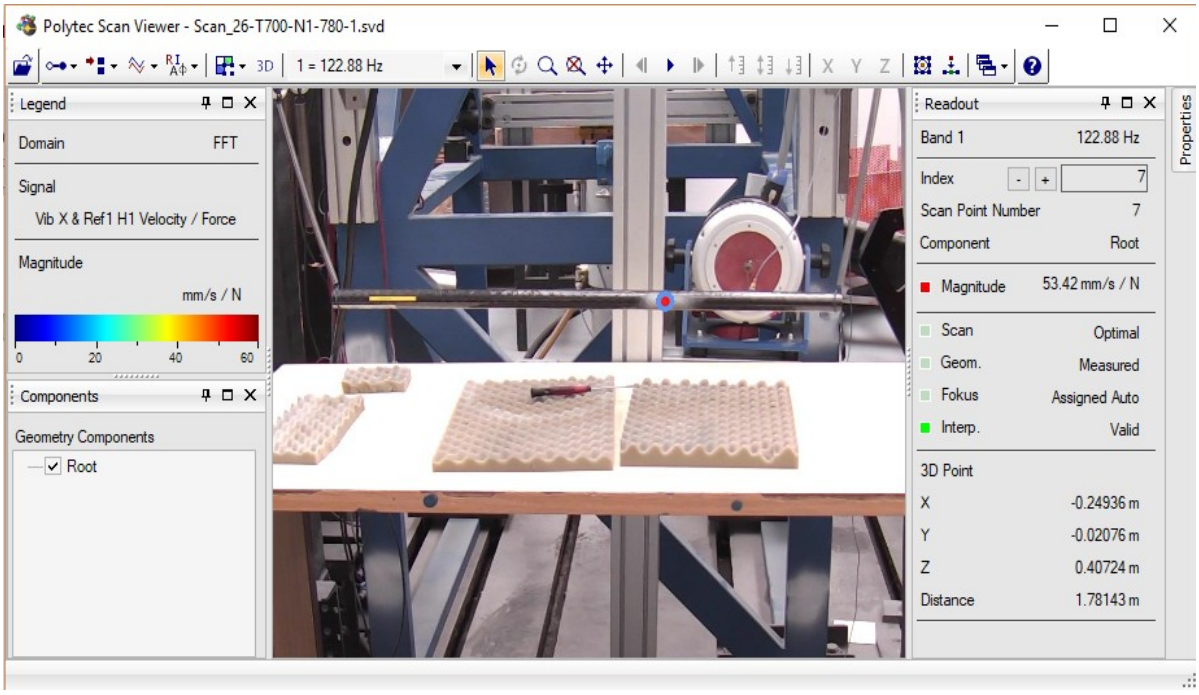
Z firmy CompoTech PLUS, spol. s.r.o. mi byla poskytnuta data z měření vibrací kompozitních struktur měřených bezkontaktním laserovým vibrometrem Polytec PSV 400 D4063. Mechanické buzení bylo zavedeno dynamickým budičem TIRA S51144-M se zesilovačem TIRA BAA 1000 a jeho skutečná síla byla měřena piezoelektrickým snímačem budící síly Brüel & Kjær 8230.

Z měření většího množství kompozitních struktur mi byla poskytnuta data z 3 druhů kompozitních struktur ve tvaru trubek a to T700, XN80 a XN90, kde každá má 4 poddruhy, které se liší typem vnitřní struktury N1, N2, P, T. Každý druh a jeho poddruh byl naměřen celkem sedm krát, čímž jsou získána data z celkem 84 měření. Každý bod obsahuje celkem 25600 naměřených hodnot na rozmezí frekvencí $f \in \langle 0.125, 3200 \rangle \, [Hz]$.

Měřená data jsou ukládána do binárního souboru navrženého firmou Polytec a běžné přístupy k načtení dat jsou zde tedy nemožné. Proto jsem napsal vlastní funkci v jazyce

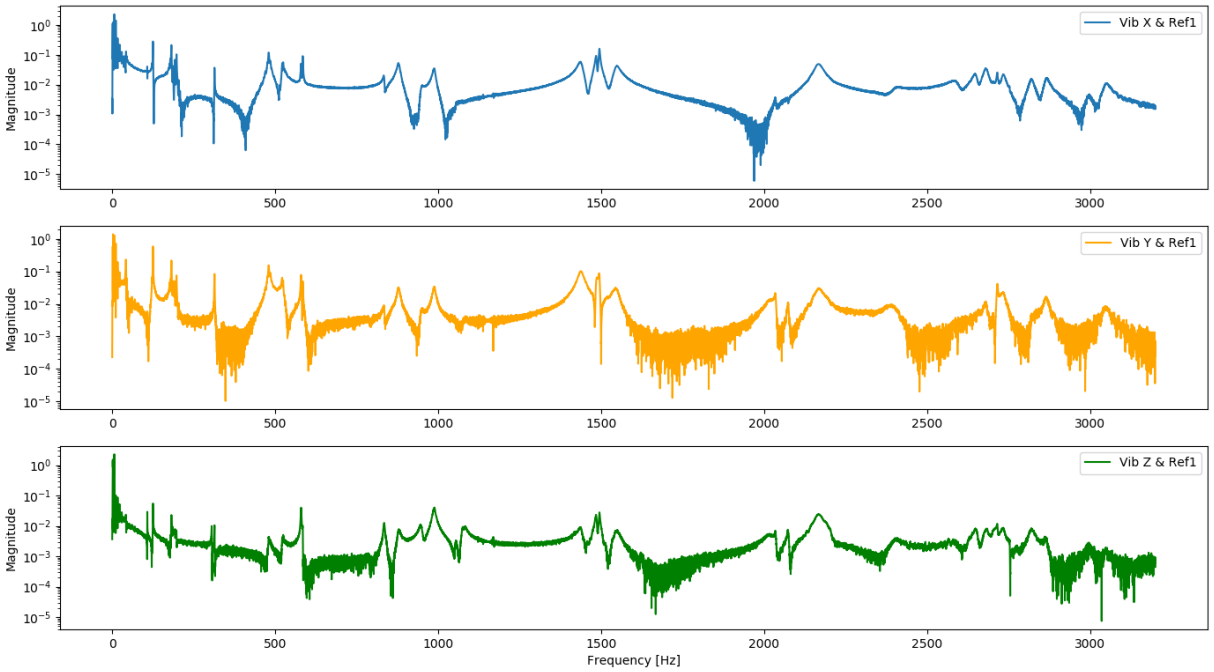
Python, která dokáže pomocí komunikace přes standard Windows COM, komponenty objektového modelu (Component Object Model), pokládat požadavky na program Polytec File Access a požadovaná data vyčíst. K zjištění požadovaných nastavení pro přístup k naměřeným hodnotám je nutné použít Polytec Scan Viewer jehož uživatelské prostředí je zobrazeno na obrázku 11 a jeho instalace je popsána v příloze 7.1.

V horní liště tohoto programu se nastavuje, jaká data budou zobrazena a zároveň je zde také zobrazeno, jaká data jsou v souboru uložena. Vpravo od nastavení zobrazovaných dat se nacházejí prvky k ovládání pořízeného obrázku a případné animace průběhu. Vlevo pod lištou nastavení se nachází legenda, která nám zobrazuje aktuální nastavení zobrazených hodnot a okno Components zobrazuje závislé soubory. Ve všech mnou analyzovaných datech se vyskytují pouze přímé hodnoty otvíraného souboru, tedy jak je zde uvedeno jako „root“. Uprostřed se nachází snímek z měření s vyznačenou pozicí měřeného bodu nebo bodů a napravo nalezneme aktuální naměřenou hodnotu dle nastavení.



Obrázek 11: Polytec Scan Viewer - uživatelské prostředí.

Vzorová ukázka průběhu vibrací v závislosti na budící frekvenci s následujícím nastavením: d*ruh - T700-N1-780; domain - FFT; channel - Vib X, Y, Z & Ref1; signal - H1 Velocity / Force; display - Magnitude* je zobrazena na obrázku 12. Hodnoty na svislé ose jsou v logaritmickém měřítku pro zřetelnější vizualizaci průběhů. Jak je již z nastavení patrné, velikost vibrací je snímána ve všech třech osách a zobrazená hodnota je velikost rychlosti vibrací vzhledem k budící síle.

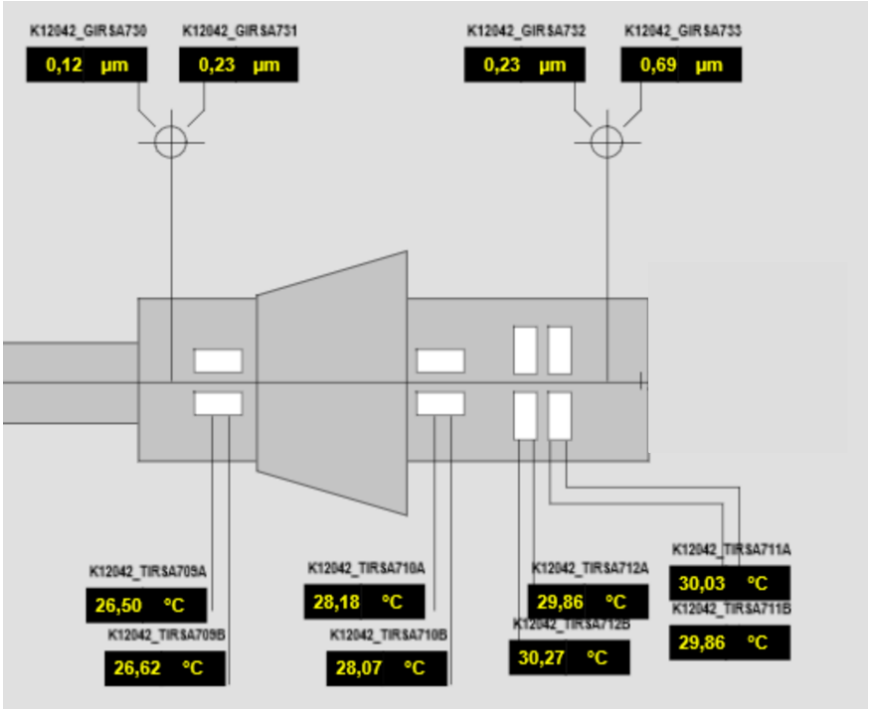


Obrázek 12: Průběh vibrací trubky T700-N1-780 na osách X, Y, Z pro různé budící frekvence.

Skript sloužící k přístupu do souborů typu SVD firmy Polytec a funkce zobrazující uvedený graf jsou uvedeny v příloze 7.3 pod polytec_svd_read.py

4.3 Howden ČKD kompresory

Z firmy Howden ČKD kompresory, spol. s.r.o. mi byla poskytnuta data z jednoho jejich turbo kompresoru, který je umístěný v Bělorusku. Analyzované hodnoty byly vybrány specialisty firmy HCKD. Jedná se o hodnoty relativních vibrací hřídele na ložiskách radiálního kompresoru a jejich přidružené teploty. Schéma analyzované části kompresoru je zobrazeno na obrázku 13. V jeho horní části se nachází hodnoty ze senzorů relativních vibrací a v jeho spodní části jsou hodnoty teplot ložisek vždy na vnitřní a venkovní straně daného ložiska.

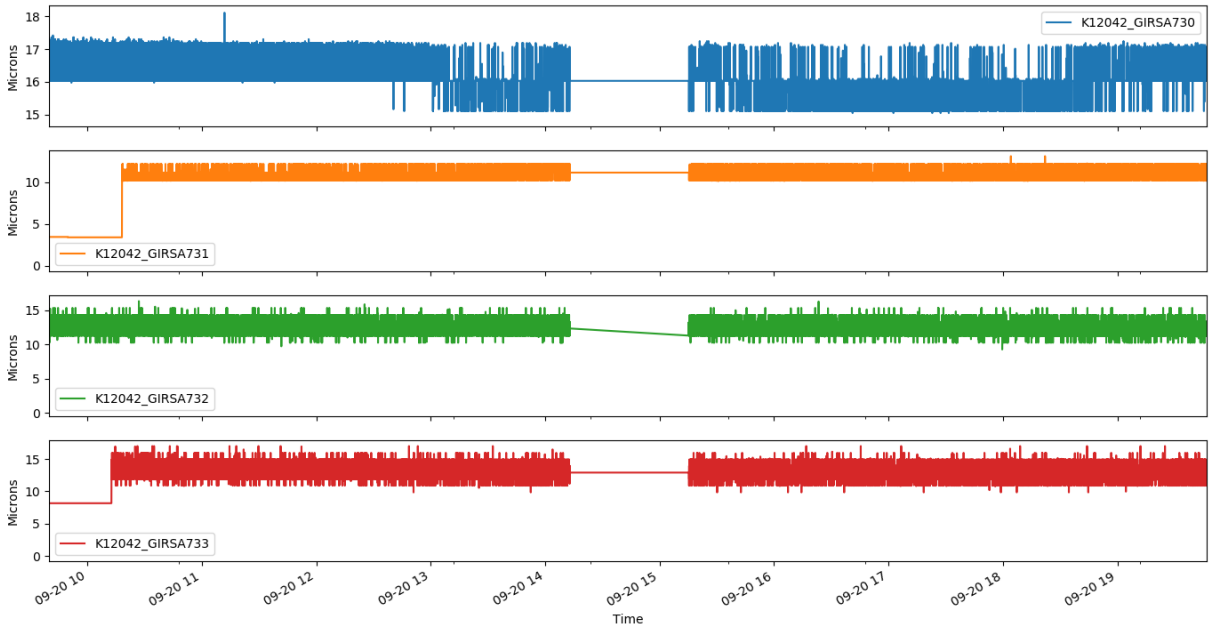


Obrázek 13: Analyzovaná část kompresoru.

Poskytnutá data byla ve formátu programu Microsoft Excel o velikosti 38MB a pro další zpracování je tento formát zcela nevyhovující, jelikož se velmi dlouho načítá a bylo nutné doplnit velké množství dat z důvodu optimalizace přenosu dat firmy HCKD Kompresory, čímž by se ještě zhoršila doba načítání, jelikož v rámci optimalizace jsou zapisovány pouze hodnoty, které se změnili. Ostatní políčka mají hodnoty typu „None“. Navrhl jsem proto skript pro převedení dat do formátu CSV, který se načítá podstatně rychleji. Každý list v Excelu odpovídal jedné skupině senzorů, proto bylo výhodné převést každý list na vlastní CSV soubor, který obsahuje vždy pouze jeden druh senzorů.

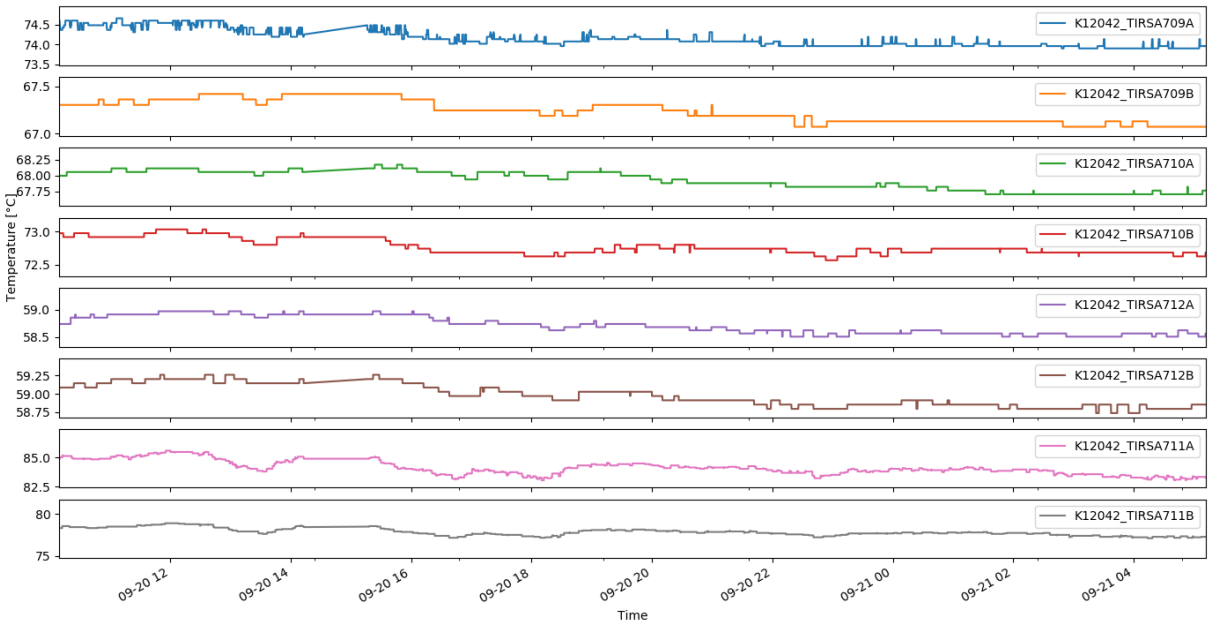
Další optimalizací dat bylo rozdělení každého monitorovaného schématu na bloky závislých hodnot. Jelikož se očekává velké množství zpracovávaných dat, jejich uchovávání ve velkých souborech by velmi zpomalilo veškeré další zpracování. Řešení s bloky závislých hodnot dočasně vyřeší tento problém a zároveň usnadní aplikace predikčních algoritmů. Dané bloky jsou vytvářeny na základě konfiguračního textového dokumentu, kde dané závislé senzory byli navrženy specialisty z HCKD.

Část průběhů analyzovaných hodnot ze senzorů relativních vibrací, což jsou GIRSA730, GIRSA731, GIRSA732 a GIRSA733 jsou zobrazeny na obrázku 14.



Obrázek 14: Průběhy hodnot senzorů relativních vibrací ložisek z kompresoru HCKD.

a část průběhů přidružených závislých teplot ze senzorů TIRSA709A, TIRSA709B, TIRSA710A, TIRSA710B, TIRSA712A, TIRSA712B, TIRSA711A, TIRSA711B jsou zobrazeny na obrázku 15.



Obrázek 15: Průběhy hodnot senzorů teplot ložisek z kompresoru HCKD.

Skript sloužící k automatickému převodu dat z poskytnutého souboru typu Excel na závislé bloky souborového typu CSV a jejich zobrazení do grafu je uveden v příloze 7.3 pod HCKD_data_preprocess.py

5 Aplikace neuronových sítí

V mé práci jsem aplikoval neuronové sítě 2.2, 2.3, 2.4 a 2.5 na základní a obvyklou problematiku datové analytiky v průmyslu jako je predikce časových řad, což je predikce která má za úkol předpovídat budoucí stav výrobního stroje a je nástrojem k prediktivní údržbě. Tuto metodu aplikace jsem otestoval na umělých datech a na datech poskytnutých společností Howden ČKD kompresory, spol. s.r.o.. Poté jsem se zabýval aplikací těchto sítí na různé struktury kompozitů z dat měřených vibrometrem Polytec za účelem otestování schopností klasifikace. Jak jsem uvedl v 1.3, součástí zpracování poruchových stavů je i klasifikace chyb z dostupných dat. Jelikož jsem ovšem neměl k dispozici žádná poruchová data, a musel bych tato data vygenerovat uměle, použil jsem dostupná data z vibrometru. Aplikace neuronových sítí za účelem klasifikace je totiž vždy velice obdobná a je tedy možné tento test považovat za ekvivalentní ke klasifikaci poruchových stavů.

5.1 Predikce časových řad

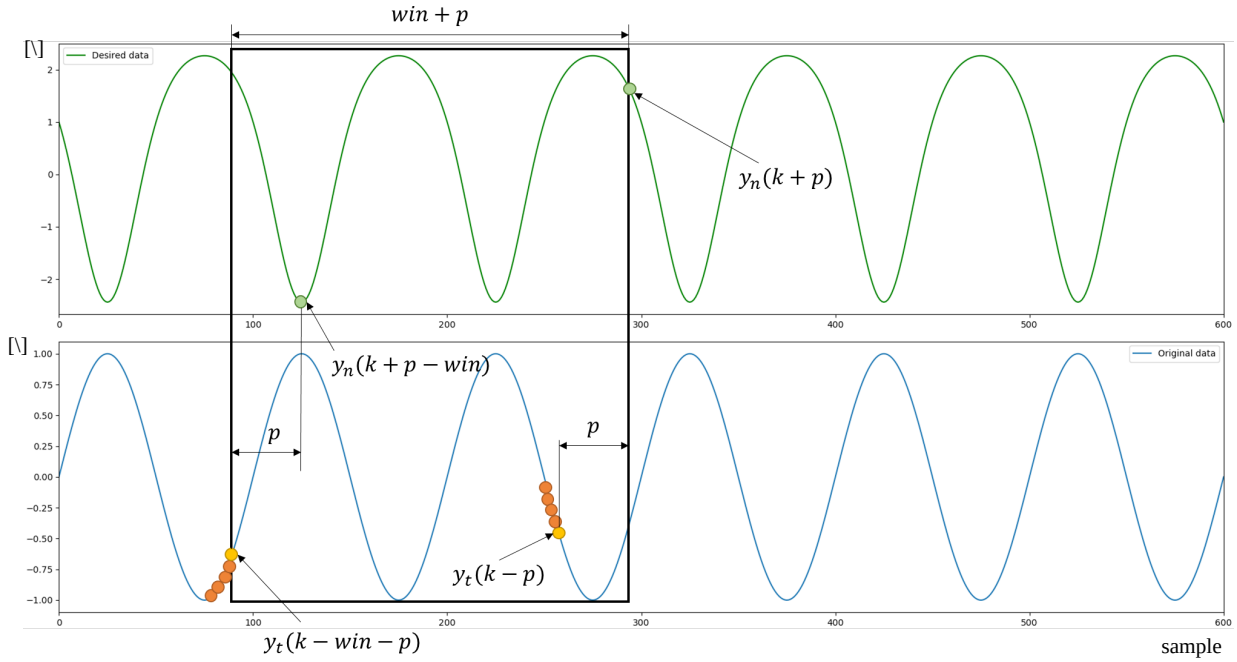
Časová řada je numerická proměnná, jejíž hodnoty závisí na čase, v němž byly získány, jde o posloupnost chronologicky uspořádaných pozorování. Časové vzorkování je pro tyto data obvykle konstantní, jelikož senzory obvykle pracují na určité frekvenci sběru dat. K predikci časových řad je nutné, aby byli neuronové sítě schopné vytvořit dostatečně přesný model systému. Tento druh predikce je základním nástrojem během metod prediktivní údržby založených na datovém modelu, jak jsem uváděl v 1.2.1 a 1.3.

5.1.1 Aplikace na umělá data

První řadu, kterou jsem se z umělých dat rozhodl predikovat je řada $\mathbf{X}_3$ z hodnot řady $\mathbf{X}_1$ jejichž průběhy jsou detailněji znázorněny na obrázku 16. Na reálných datech může tento druh predikce představovat dopočet hodnot jednoho senzoru, který je korelovaný se senzorem jiným.

Jelikož víme že tyto řady jsou na sobě závislé pomocí exponentu, lepší výsledek z neuronových sítí typu HONU by měla poskytnout síť QNU nežli LNU. U sítí typu ELM a LSTM jsou také očekávány velmi dobré výsledky za předpokladu užití dostatečného počtu neuronů ve skryté vrstvě u ELM nebo buněk u LSTM.

K aplikování metody predikce jsem zvolil metodu plovoucího okna, jehož princip je znázorněn na obrázku 16. K této metodě je nutné zvolit šířku okna ($window$) a délku predikce (p). Tímto dostanu vstupní matici $\mathbf{X^M}$, jejíž řádky reprezentují vstupní vektor do neuronu a dále mám vektor cílových (výstupních) dat y_t. Vstupní matice má rozměr $\mathbf{L} \times n$, kde n je velikost vstupního vektoru do neuronu včetně biasu a $\mathbf{L}$ je rozměr vektoru cílových dat. Metoda plovoucího okna nám také umožňuje přetrénovat neuronové sítě vždy po určitém počtu vzorků, avšak na použitých umělých datech není přetrénování nutné.



Obrázek 16: Průběhy časových řad $\mathbf{X}_1$ a $\mathbf{X}_3$ (Obrázek 10, rovnice (27) (29)) a princip predikce metodou klouzajícího okna. Zvolí se šířka okna ($win = window$) a délka predikce (p). Žlutý a oranžové kruhy představují konkrétní hodnotu ve vstupní matici ze které se vypočítává hodnota predikovaná.

Vstupní vektor do neuronu, který je součástí vstupní matice, obsahuje vstupní data a jejich nedávnou minulost a vypadá tedy následovně

$$\mathbf{X^M} = \begin{bmatrix} 1 & y_t(k) & y_t(k-1) & \cdots & y_t(k-n+1) \\ 1 & y_t(k-1) & y_t(k-2) & \cdots & y_t(k-n+2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & y_t(k-window) & y_t(k-window-1) & \cdots & y_t(k-window-n+1) \end{bmatrix} \quad (33)$$

a výstupem z neuronu pro dávkové učení je vektor vypočítaných hodnot

$$\mathbf{y_n} = [y_n(k+p) \quad y_n(k+p-1) \quad \cdots \quad y_n(k+p-win)]. \quad (34)$$

Pojem klouzající okno zde sice může vzhledem k užití dávkového učení znít jako zavádějící, jelikož se vstupní matice $\mathbf{X^M}$ učí na celé okno bez klouzání, avšak princip aplikace tato metoda popisuje zcela správně.

Pro mou aplikaci jsem zvolil počet zpožděných hodnot $n = 2$, což je vzhledem k periodicitě daných dat dostatečný počet předchozích hodnot, a i dle provedeného testování vyšší hodnoty neměli vliv na přesnost neuronové sítě vliv. K natrénování LNU a QNU jsem použil algoritmus Levenberg-Marquardt s nastavením parametrů je dle tabulky 2.

Tabulka 2: Parametry neuronových sítí LNU a QNU.

Neuronová síť	LNU	QNU
Učící algoritmus	LM	LM
Neurony	3	6
Predikce	+10 vzorků	+10 vzorků
Rychlost učení	2	2
Trénovací okno	200 vzorků	200 vzorků
Epoch trénování	30	30

Trénovací časy obou neuronových sítí byly vzhledem k malému množství dat zanedbatelné. Avšak déle trvalo natrénovat síť QNU, což je dáno vyšším počtem neuronů.

Pro aplikaci neuronové sítě typu ELM byla použita knihovna hpelm od autorů uvedených v [33]. Parametry neuronové sítě byli nastaveny obdobně jako pro sítě LNU a QNU jak je vidět v tabulce 3. Natrénování této sítě bylo také velmi rychlé.

Tabulka 3: Parametry neuronové sítě typu ELM.

Neuronová síť	ELM
Aktivační funkce	tanh
Neurony	24
Predikce	+10 vzorků
Trénovací okno	200 vzorků

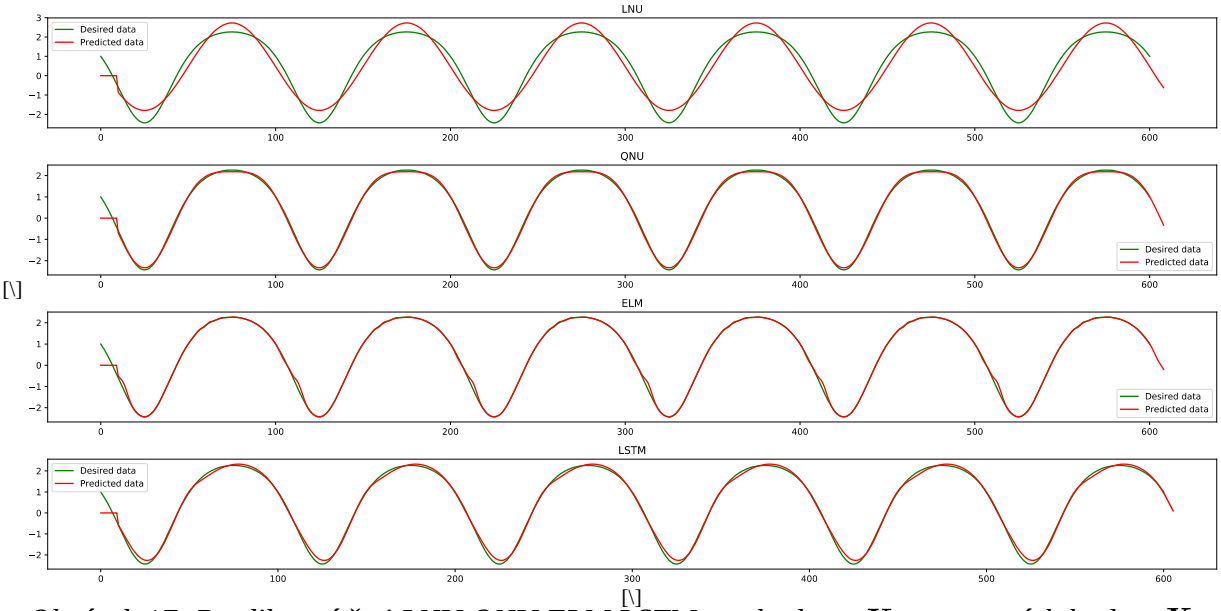
V případě neuronové sítě typu LSTM jsem aplikoval knihovnu Keras, která je dostupná z [34]. Parametry neuronové sítě byly nastaveny dle tabulky 4. Tato nastavení se liší od předchozích sítí, což je dáno značně komplikovanější strukturou jednotek této sítě.

Tabulka 4: Parametry neuronové sítě typu LSTM.

Neuronová síť	LSTM
Buňky	40
Predikce	+10 Vzorků
Trénovací okno	200 Vzorků
Dropout	0.2
Epoch trénování	200

Jednotky typu LSTM obvykle pracují ve větších celcích s větším množstvím historických hodnot, proto jsem po náležitém otestování zvolil nastavení na 40 neuronů jako optimální řešení s přihlédnutím na trénovací čas a dostatečně přesné výsledky. Komplikovanost celé struktury LSTM také vyžaduje podstatně vyšší počet epoch trénování a ze zvolených metod se jedná o nejdéle trénovanou neuronovou síť. Vysoký počet trénovacích epoch by mohl způsobit přetrénování této sítě, proto jsem použil funkci knihovny Dropout, která v každé epoše náhodně nastaví určitý počet hodnot ve vstupním vektoru na 0. Další změnou ke zlepšení fungování sítě LSTM bylo přidání historických hodnot z $n = 2$ na $n = 10$, jelikož na rozdíl od předchozích sítí je dokázala tato síť využít a zlepšit její prediktivní schopnosti.

Výše uvedené změny v testovacím nastavení jsem provedl po otestování podobných nastavení, které byli použity pro trénování sítí LNU, QNU a ELM, jelikož tato dříve použitá nastavení vedla na velké chyby celé této sítě což je dáno její komplexností.



Obrázek 17: Predikce sítěmi LNU,QNU,ELM,LSTM pro hodnoty $\mathbf{X_3}$ z vstupních hodnot $\mathbf{X_1}$.

Jak je vidět na obrázku 17, předpoklad lepšího výsledku sítě QNU než LNU byl správný, což je dáno jednak vyšším počtem neuronů, tak exponenciální charakteristikou požadovaných dat a sítě. Je patrné, že síť typu LNU nedokáže dostatečně uspokojivě vystihnout exponenciální charakteristiku, pokud jí neobsahují zdrojová data, zatímco síť QNU to zvládá velice dobře. Síť typu ELM nemá tak hladký průběh jako síť QNU ale charakteristiku predikované funkce zvládá vystihnout s malou chybou. Síť typu LSTM má hladký průběh, obdobně jako QNU, ale přesnost predikce je nižší. Až na LNU dokážou všechny sítě predikovat charakteristický průběh požadované funkce.

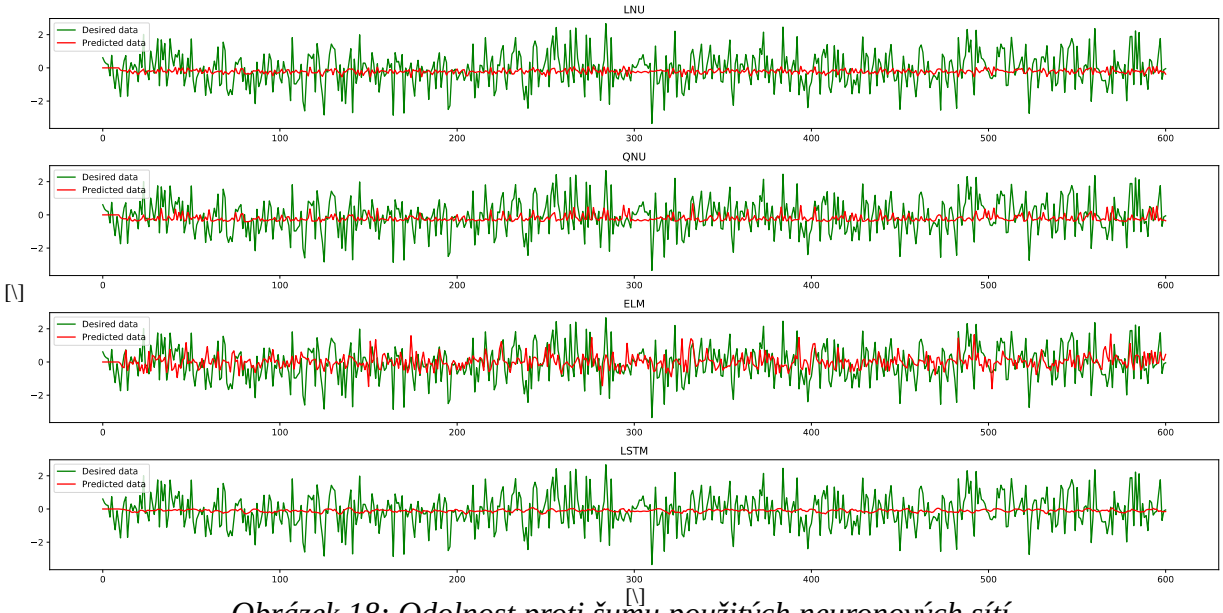
Hodnoty průměrných chyb kvadrátu predikce (Mean square error, MSE) vyjádřené jako

$$MSE = \frac{1}{N}\sum_{k=0}^{N}(y_t - y_n)^2, \tag{35}$$

kde N je počet všech vzorků, y_t je reálná hodnota výstupu a y_n je vypočtený výstup neuronové sítě. Hodnoty MSE jsou následující: $LNU = 0.151411$, $QNU = 0.003992$, $ELM = 0.0358082$ a $LSTM = 0.0190867$.

Obdobný postup predikce by byl aplikován i na data $\mathbf{X_5}$. Jedinou nutnou změnou by bylo proložit tuto funkci přímkou k zachycení stoupajícího trendu čímž bychom získali obyčejný průběh funkce $sinus$, který lze pomocí neuronových sítí lehce vymodelovat.

K ověření, že neuronové sítě opravdu modelují daný systém a nesledují pouze vstupní hodnoty se obvykle používá otestování na predikci šumu, zde se jedná o hodnoty z $\mathbf{X_2}$ nebo $\mathbf{X_4}$, které se přivedou na vstup i výstup neuronových sítí, jak je ukázáno na obrázku 18 pro



Obrázek 18: Odolnost proti šumu použitých neuronových sítí.

data z $\mathbf{X_2}$. Zároveň se tímto otestuje odolnost jednotlivých neuronových sítí proti šumu na vstupních datech. Nastavení všech sítí bylo ponecháno stejné jako v případě předchozí predikce.

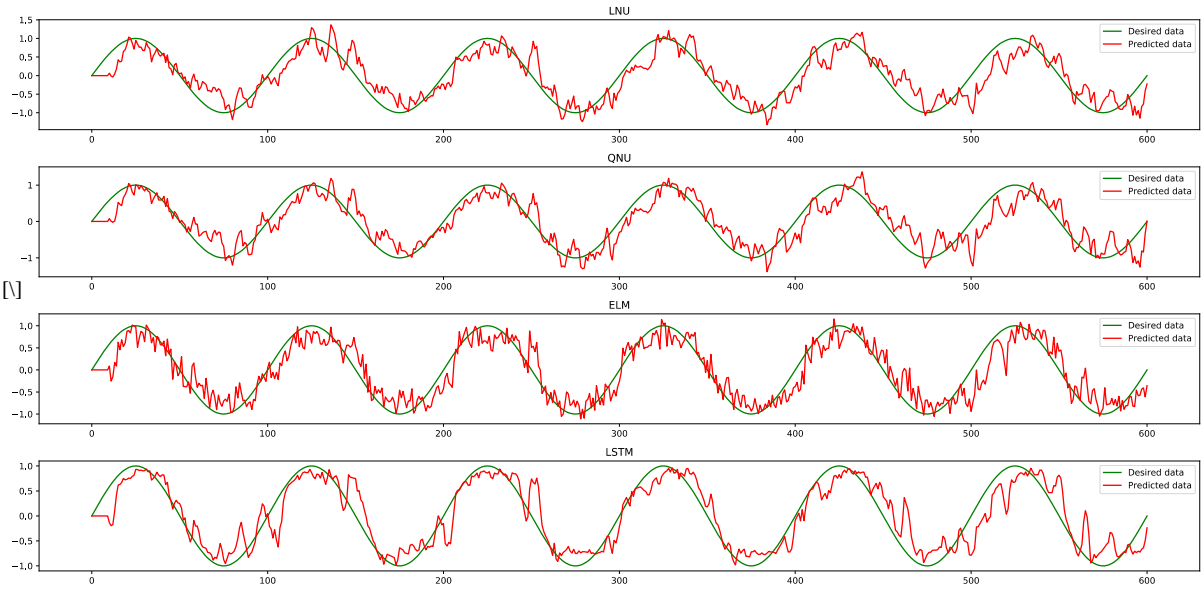
Provedením tohoto testu jsem potvrdil, že žádná z aplikovaných sítí nesleduje pouze vstup, ale opravdu vytváří model systému. Z testovaných sítí má nejlepší výsledek odolnosti proti šumu typ LSTM, což je dáno možností regulace dat pomocí bran. Sítě typu LNU a QNU dosahují navzájem obdobné odolnosti na šum, avšak nedokážou ho tak dobře potlačit jako síť z jednotek LSTM. Nejhoršího výsledku dosahuje síť druhu ELM, jejíž výstup je značně zašuměný, což je dáno vysokým počtem neuronů s náhodnými váhami ve vstupní vrstvě neuronové sítě, jelikož tyto váhy mohou šum ještě více umocnit. Velikost chyb MSE nemá v tomto případě smysl sledovat, jelikož šlo především o potlačení šumu.

Jako poslední test k ověření schopností vytváření modelů systému jsem zvolil predikci funkce $\mathbf{X_1}$ z dat funkce $\mathbf{X_6}$ jelikož tato data jsou zatížena šumem, jedná se kombinaci dvou obvyklých cílů v datové analytice, a to vytvoření modelu a potlačení šumu. Provedl jsem změnu v nastavení na více historických hodnot i pro sítě HONU a ELM, tedy $n = 10$, jelikož v tomto případě jejich užití má již pozitivní dopad na schopnosti jejich predikce. Ostatní změny jsou uvedeny v tabulce 5.

Tabulka 5: Nastavení neuronových sítí k predikci $\mathbf{X_1}$ z $\mathbf{X_6}$.

Neuronová síť	LNU	QNU	ELM	LSTM
Neurony	11	66	100	66
Predikce	+10	+10	+10	+10
Trénovací okno	400	400	400	400
Epoch trénování	30	30	konv.	100

Všem sítím jsem zvedl počet trénovacích dat na 400 s cílem lepšího naučení neuronových sítí. Stejně tak jsem zvedl počet neuronů u sítí ELM a LSTM. Schopnosti predikce těchto neuronových sítí jsou vedeny na obrázku 19.



Obrázek 19: Predikce $\mathbf{X_1}$ z dat zatížených šumem $\mathbf{X_6}$.

Z výsledků predikce dat zatížených šumem je patrné, že všechny sítě jsou schopné vytvořit model který lze využít k predikci i na datech zatížených šumem. Chyby neuronových sítí, MSE, dle vzorce (35), jsou $LNU = 0.106753$, $QNU = 0.095522$, $ELM = 0.090784$ a $LSTM = 0.0785$.

Největší průměrné utlumení hodnot šumu poskytuje síť LSTM, avšak zároveň má také nejvyšší hodnoty absolutních chyb. Přesto dokáže tento systém predikovat s nejmenší chybou. Ze sítí HONU je opět lepší síť typu QNU nežli LNU. Síť QNU má také menší hodnoty absolutních chyb, než má síť LSTM, ale potlačení šumu není tak dokonalé. Neuronová síť typu ELM má dokonce menší hodnotu MSE než QNU, ale je silně zatížena šumem vstupních dat z důvodů, které byli zmíněny dříve.

5.1.2 Reálná data z Howden ČKD kompresory

Jelikož se jedná o data reálná, je nutné tato data nejdříve upravit do optimálnější podoby. K demonstraci predikce jsem využil data ze senzoru K12042_GIRSA732, který měří relativní vibrace hřídele kompresoru. Princip predikce jsem ponechal obdobný jako u umělých dat, aby bylo možné otestovat podobné nastavení na značně komplikovanějším a neznámém modelu. Tento senzor jsem zvolil, jelikož jako jediný má z poskytnutých dat hodnoty od začátku datového souboru a budu tedy moci využít celou jejich délku.

Na druhé ukázce zpracování reálných dat, která má ověřit schopnosti vytvořil model mezi dvěmi korelovanými hodnotami, jsem zvolil hodnoty ze senzorů K12042_TIRSA712A a

K12042_TIRSA712B, což jsou hodnoty teplot jednoho ložiska na vnitřní a vnější straně. Je zde tedy jistota, že hodnoty budou dosahovat vysoké korelace.

Prvním krokem během zpracování těchto dat bylo provedení převzorkování. Původní data obsahují hodnoty ze senzorů za každou půl vteřinu s velmi malými změnami, které se jeví spíše jako šum než reálná změna. Jako optimální vzorkování mi během mých testů vyšlo vytvořit průměr z 30 vzorků, čímž také začne být patrná charakteristika průběhu vibrací. Z původních 65535 záznamů jsem tedy vytvořil 2183.

V dalším kroku jsem provedl úpravu dat metodou Z-Score dle vzorce

$$Z = \frac{x_i - \overline{x}}{s}, \tag{36}$$

kde x_i je aktuální hodnota, $\overline{x}$ je průměrná hodnota a s je směrodatná odchylka vypočítaná dle

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}. \tag{37}$$

Bohužel se ale tato úprava ukázala jako nevhodná, a proto jsem na místo ní aplikoval metodu normalizace dle vzorce

$$z_i = \frac{x_i - min\,(\mathbf{X})}{max\,(\mathbf{X}) - min\,(\mathbf{X})}, \tag{38}$$

kde z_i je normalizovaný člen dle x_i a $\mathbf{X}$ je vstupní vektor. Tato úprava převede data na hodnoty od 0 do 1 a v datové analytice je hojně využívána.

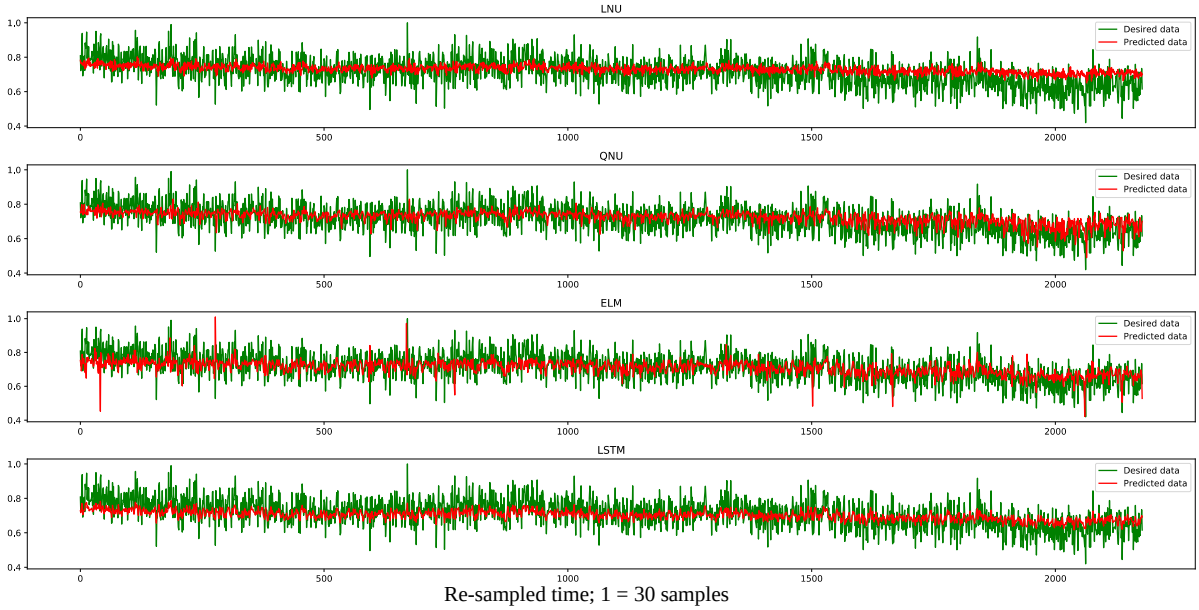
Neuronovým sítím byli poskytnuty 3 historické hodnoty. Ostatní nastavení jsou uvedena v tabulce 6.

Tabulka 6: Nastavení neuronových sítí pro data vibrací z HCKD kompresory.

Neuronová síť	LNU	QNU	ELM	LSTM
Neurony	4	10	50	66
Predikce	+1	+1	+1	+1
Trénovací okno	1500	1500	1500	1500
Aktivační funkce	lin	lin	sigm	sigm
Dropout	-	-	-	0.02
Rychlost učení	7	0.07	-	-
Epoch trénování	30	30	konv.	100

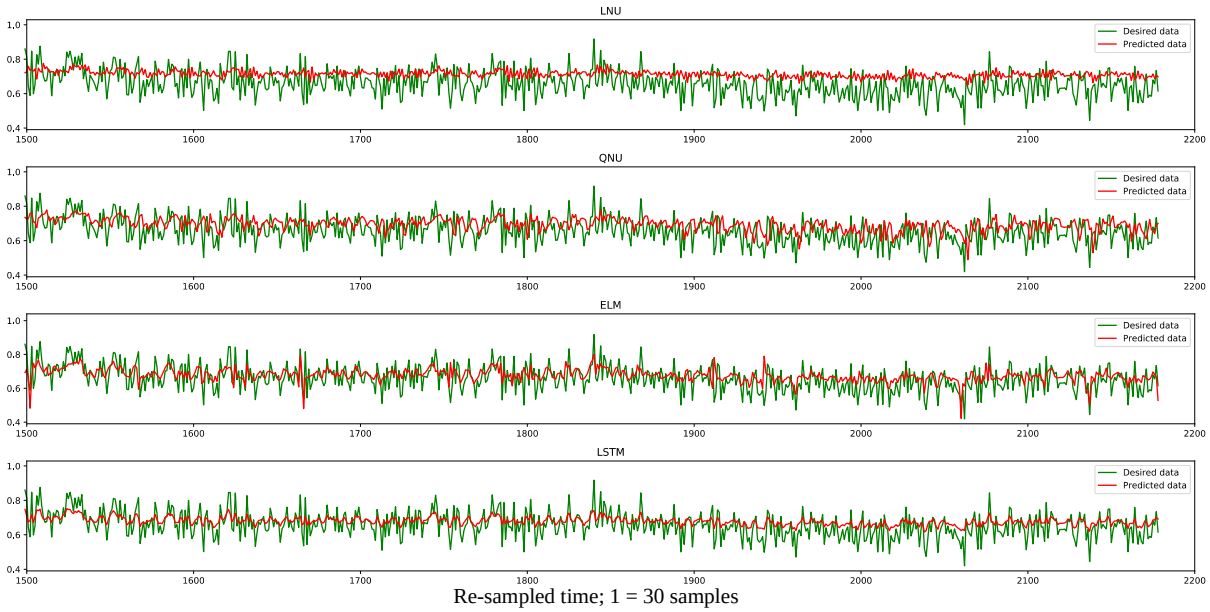
Z aplikovaných sítí trvalo opět nejdéle natrénovat síť typu LSTM a nejrychlejší bylo trénování sítě ELM.

Při pohledu na obrázek 20 vidíme, že hodnoty vibrací mají klesající charakter, který není zcela modelován ani jednou z použitých sítí. Síť QNU sice lépe predikuje i nižší hodnoty na testovacím průběhu, avšak po většinu času setrvává na úrovni pomyslné přímky.



Obrázek 20: Predikce vibrací +1 vzorek, celá data včetně trénovací sady.

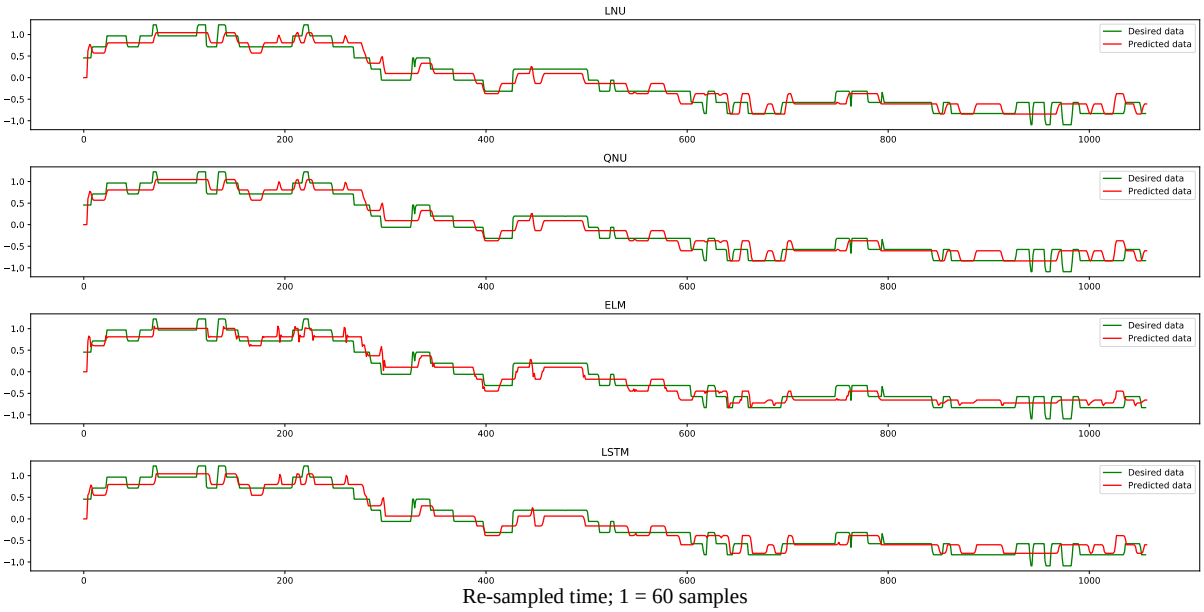
Při bližším pohledu na testovací část, což je část, na které nebyly neuronové sítě trénovány (1501 - 2150), zobrazenou na obrázku 21 zjistíme, že nejvěrohodnější model poskytuje síť LSTM, jelikož charakteristický průběh téměř vždy odpovídá požadované hodnotě, pouze nemá požadovanou velikost, což může být způsobeno utlumením šumu typickým pro LSTM. Ke zlepšení výsledků predikce vibrací by bylo nutné provést pokusy s různým datovým předzpracováním. Případně také uvažovat i jiné vstupní hodnoty, které by mohly vibrace ložisek ovlivnit. Na tomto obrázku je také vidět, že relativní vibrace mají klesající charakter a jsou tedy korelované s klesajícími teplotami na obrázku 22. Při dalším zpracování by tedy mohlo být výhodné kromě vibrací samotných uvažovat i teplotu jako součást vstupních dat.



Obrázek 21: Predikce vibrací +1 vzorek, testovací část.

Síť typu ELM lépe vystihuje velikost hodnot než síť LSTM, avšak dodržet charakteristický průběh se jí vždy nedaří. Obdobných výsledků dosahuje i síť QNU, která je ovšem zatížena ještě větší chybou. Vylepšení sítí HONU by mohlo nastat po aplikování jiného hodnocení chyby a tím úpravy průběhu učení.

K predikci hodnot teploty ze senzoru K12042_TIRSA712B jsem použil data ze senzoru K12042_TIRSA712A jejich průběh je uveden na obrázku 15. Rozdílem v nastavení je užití metody Z-Score místo normalizace a zvýšení počtu vzorků v jedné průměrné hodnotě z 30 na 60 hodnot, jelikož teplota se v čase nemění nikterak výrazně a poskytnutá data mají často velmi dlouhé oblasti bez změny teploty. Tyto dlouhé úseky bez změn teplot jsou dány nízkou citlivostí použitých senzorů, anebo také nevhodný převod z kompresoru do databáze HCKD a je tedy nutné zmínit, že tato data nejsou příliš vhodná pro zpracování pomocí neuronových sítí. Bohužel ovšem nemám k dispozici data jiná, a i takováto data se v průmyslu vyskytují, proto je výhodné ověřit, jak si s tímto problémem testované sítě poradí i přes očekávané špatné výsledky.

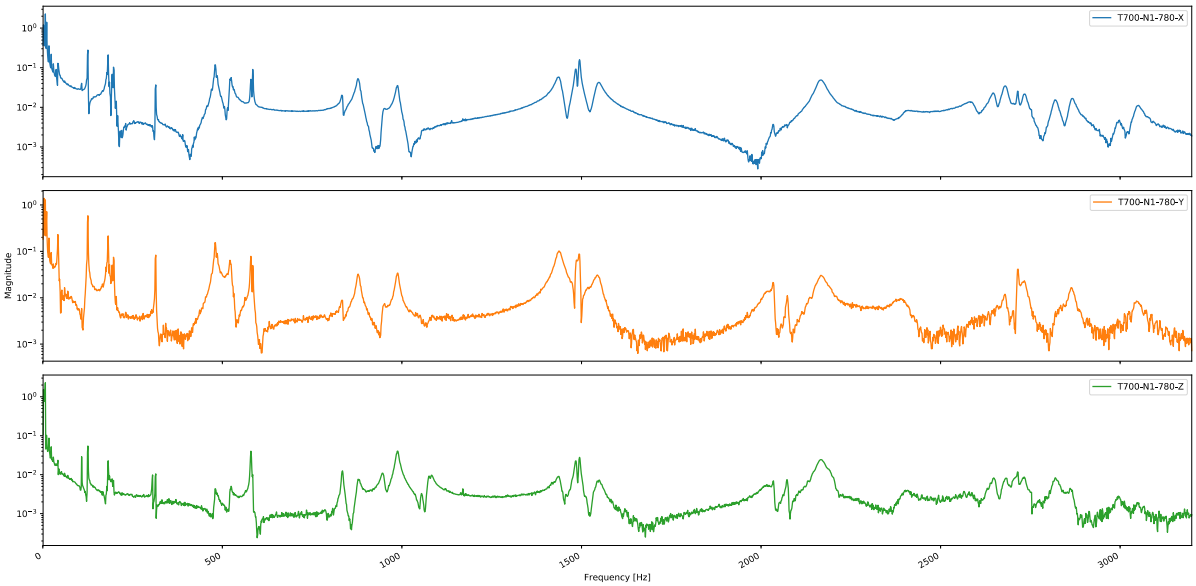


Obrázek 22: Predikce (+1 vzorek) teploty senzoru 712B z hodnot 712A za použití z-score metody.

Výsledky, které jsou k vidění na obrázku 22 předčily má očekávání. Ačkoliv jsou data takto nevhodná, neuronové sítě stále dokáží relativně dobře vymodelovat závislost mezi dvěma senzory s uspokojujícími výsledky. Chyba MSE všech sítí se pohybuje okolo hodnoty 0.027, avšak je vidět, že ani jedna testovaná síť nedokáže potlačit náhodné skoky teplot ze vstupního senzoru bez dalšího datového zpracování.

5.2 Klasifikace kompozitních struktur

K otestování schopností klasifikace jednotlivých neuronových sítí jsem použil data z vibrometru Polytec, jejichž popis a způsob přístupu k nim je uveden v 4.2. Konkrétně jsem se zabýval klasifikací 3 druhů kompozitních struktur, a to T700-N1, XN80-N1 a XN90-N1. Cílem této klasifikace tedy je, aby neuronová síť s dostatečnou přesností určila, o jaký typ trubky se jedná v závislosti na poskytnutých datech. Jelikož délka měřených dat je 25600 vzorků a tím pádem by bylo velice obtížné, a hlavně časově náročné, neuronové sítě na takovéto množství data natrénovat, opět jsem tedy provedl jejich převzorkování. Tentokrát jsem použil maximum z deseti vzorků a při porovnání průběhů na obrázku 12 s převzorkovaným průběhem na obrázku 23 je patrné, že tímto krokem se značně snížila hodnota šumu na některých frekvencích, zatímco charakteristický průběh zůstal zcela zachován či je dokonce lépe pozorovatelný.



Obrázek 23: Převzorkovaný průběh vibrací kompozitu T700-N1-780.

Jak je z obrázku 23 vidět, každou odezvu daného druhu kompozitu v závislosti na frekvenci lze složit ze tří hodnot, které v tomto případě reprezentují vibrace ve směrech X, Y a Z a tyto hodnoty jako takové musí dát vždy unikátní kombinaci, díky které je může neuronová síť rozpoznat a náležitě klasifikovat. Jako problematické jsou očekávané oblasti frekvencí, kde jiné struktury budou mít velmi podobný průběh vibrací a nelze je tedy jednoznačně rozeznat. Další případnou komplikací této metodiky může být prohození os X, Y, Z, což by mohlo bez náležitého předzpracování vést na velmi slabé výsledky klasifikace.

Jako vhodné datové předzpracování jsem po náležitém otestování zvolil metodu Z-Score dle vzorce (36), jelikož po její aplikaci dosahovali aplikované neuronové sítě nejlepších výsledků klasifikace.

Jako výstupní data jsem vytvořil sloupce, kde každý sloupec reprezentuje jednu osu vibrací a každý jeden řádek reprezentuje hodnotu vibrace pro danou frekvenci v dané ose. Na výstupu z klasifikátorů byly v tomto případě očekávány 3 možné výsledky, proto jsem vytvořil jednotkovou matici o velikosti 3×3

$$\mathbf{Y_n} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{39}$$

kde první řádek byl vždy přiřazen hodnotám z kompozitu T700-N1-780, druhý řádek k hodnotám kompozitu XN80-N1-780 a třetí řádek k hodnotám kompozitu XN90-N1-780.

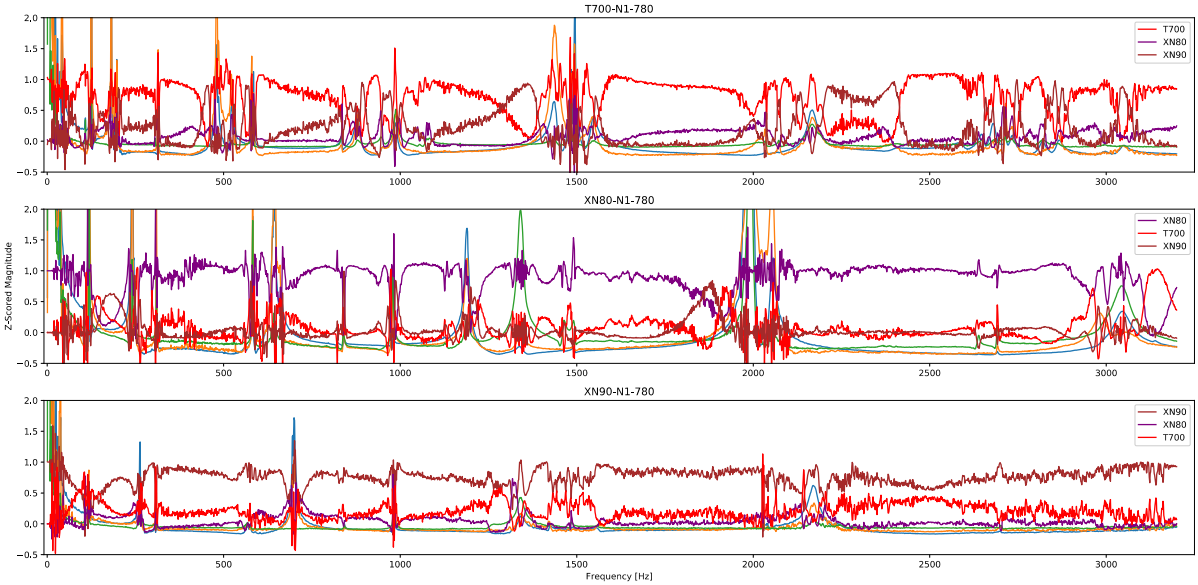
5.2.1 Aplikace sítě ELM

Opět jsem využil knihovnu hpelm od autorů uvedených v [33]. Tentokrát v konfiguraci klasifikátoru s nastavením na vícero druhů ke klasifikaci. Jelikož je nutno klasifikovat 3 druhy, výstupem z ELM jsou pro každou kombinaci vstupu tři hodnoty, kde každý sloupec reprezentuje pravděpodobnost jednoho druh kompozitu, jak již bylo vysvětleno dříve. V případě ELM pro natrénování výstupu na tři hodnoty potřebujeme 3 výstupní neurony. Čímž se nám také ztrojnásobí počet vah, které musíme trénovat. Síť ELM byla nastavena dle tabulky 7.

Tabulka 7: Nastavení ELM pro klasifikaci.

Neuronová síť	ELM
Neurony	312 x 3
Druh klasifikace	multi-label
Aktivační funkce	sigm

Z obrázku 24 je patrné, že předpoklad možnosti klasifikace na základě kombinace vibrací ve směrech X, Y, Z byl správný a tento přístup je možný. Problematickými body jsou dle očekávání frekvence, kde se signály začínají překrývat a neuronová síť zde tedy nemůže jednoznačně určit, kam dané hodnoty patří.



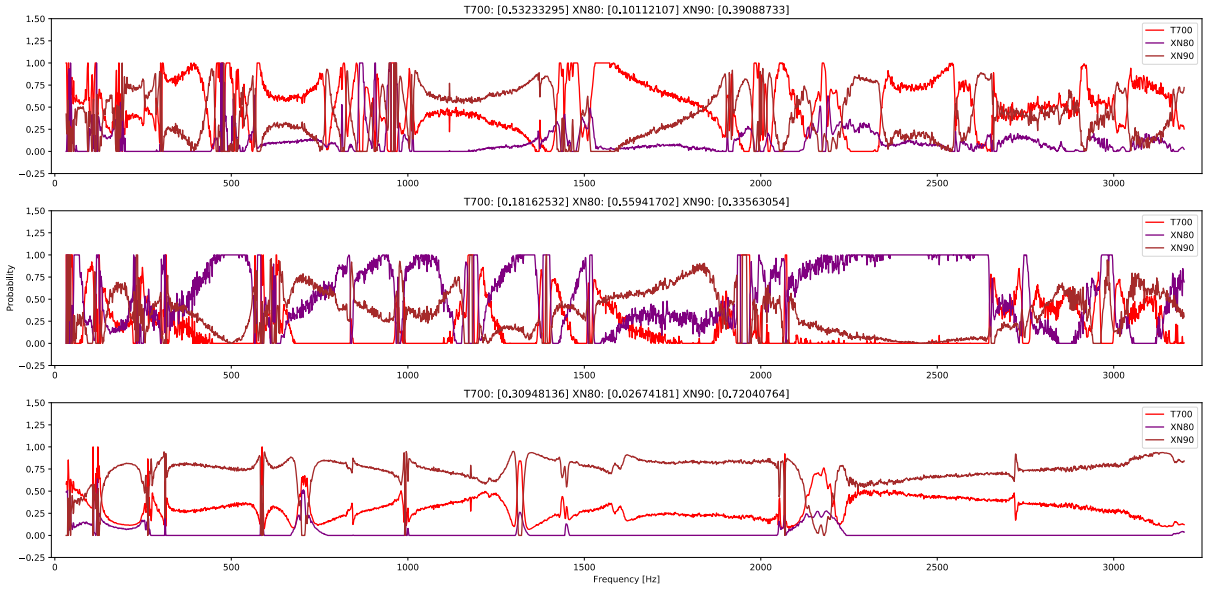
Obrázek 24: Trénovací data klasifikace tří druhů kompozitů pomocí sítě ELM. Hodnoty pravděpodobností klasifikace jsou vyjádřeny červeně pro T700-N1, fialově pro XN80-N1 a hnědě pro XN90-N1.

Úspěšnosti klasifikace neuronové sítě ELM pro jednotlivé kompozity vzhledem k frekvenci jsou uvedeny v tabulce 8. Tato úspěšnost reprezentuje, kolik procent z celého frekvenčního pásma bylo možné využít k úspěšné klasifikaci. Tato úspěšnost by mohla být ještě více zvýšena dalším datovým předzpracováním, jako například vyloučení frekvencí, na kterých nastává překrytí hodnot, anebo užitím více trénovacích vzorků k pokrytí širšího množství možných kombinací.

Tabulka 8: Úspěšnost klasifikace sítě ELM.

T700-N1-780	81,48%
XN80-N1-780	90,55%
XN90-N1-780	94,67%

Síť natrénovanou na data zmíněná dříve jsem použil k rozpoznání druhu kompozitu z dalšího měření abych otestoval schopnosti detekce i na takto řídká trénovací data. Výsledek je uveden na obrázku 25. V tomto případě jsem opět využil celé měřené frekvenční pásmo a procentuální pravděpodobnost daného kompozitu jsem vypočetl jako součet příslušného sloupku daného kompozitu podělený celkovým počtem vzorků.



Obrázek 25: Pravděpodobnost klasifikace kompozitů do tří tříd pro celé frekvenční spektrum pomocí sítě ELM. Červená – T700-N1, fialová – XN80-N1, hnědá XN90-N1.

Všechna nová data se neuronové síti povedlo úspěšně klasifikovat. Úspěšnosti neuronové sítě v klasifikaci jednotlivých kompozitů jsou pro přehlednost uvedeny v tabulce 9.

Tabulka 9: Úspěšnost klasifikace kompozitů dle sítě ELM na nově skenované sadě.

Graf	Kompozit	Pravděpodobnost
1	T700-N1	53,23%
2	XN80-N1	55,94%
3	XN90-N1	72,04%

Dle očekávání byli problematické body opět v oblastech překrytí hodnot. Další zhoršení pravděpodobnosti úspěšné klasifikace byla změna v aktuální konfiguraci vnitřní struktury kompozitu, která by se dala potlačit natrénováním sítě na více měřeních než pouze na jednom. Nicméně vzhledem k množství trénovacích dat, jsou výsledky více než uspokojivé a metodika i síť mají dobré předpoklady ke zlepšení.

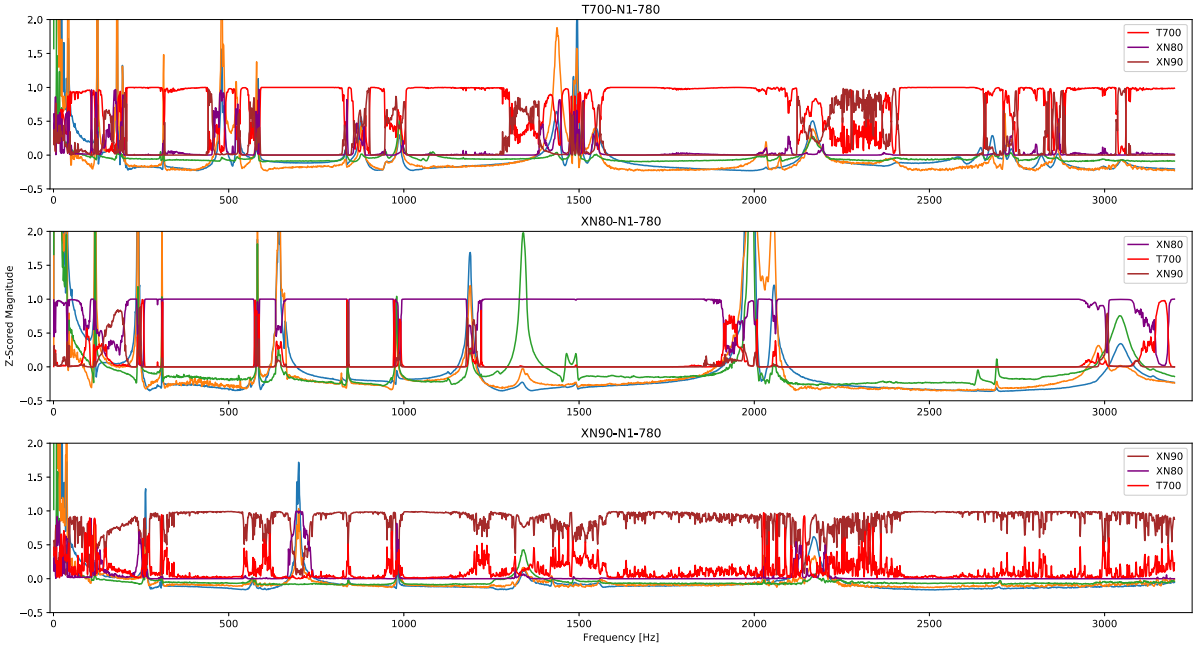
5.2.2 Aplikace sítě LSTM

K aplikaci sítě LSTM jsem opět využil knihovnu Keras obdobně jako v případě regrese. Datová příprava zůstala zachována stejná jako v případě sítě ELM. Síť LSTM byla nastavena dle tabulky 10. Největší změnou proti předchozím implementacím je užití více vrstev, čímž by se měli zlepšit klasifikační schopnosti celé této sítě. Avšak vytváří se zde určitá nejistota vhledem k rozšíření oblasti skryté vrstvy, kterou lze jen obtížně analyzovat.

Tabulka 10: Nastavení LSTM pro klasifikaci.

Neuronová síť	LSTM
Neurony	100
Vrstev	3
Epoch	20
Dropout	0.1
Aktivační funkce výstupu	sigmoid

Z obrázku 26 je opět patrné, že síť dokáže na většině frekvencí rozpoznat, o který kompozit se jedná. Na rozdíl od ELM má podstatně delší úseky, které rozpoznává s podstatně vyšší pravděpodobností. Což je dáno její schopností naučit se i velice dlouhé vzory.



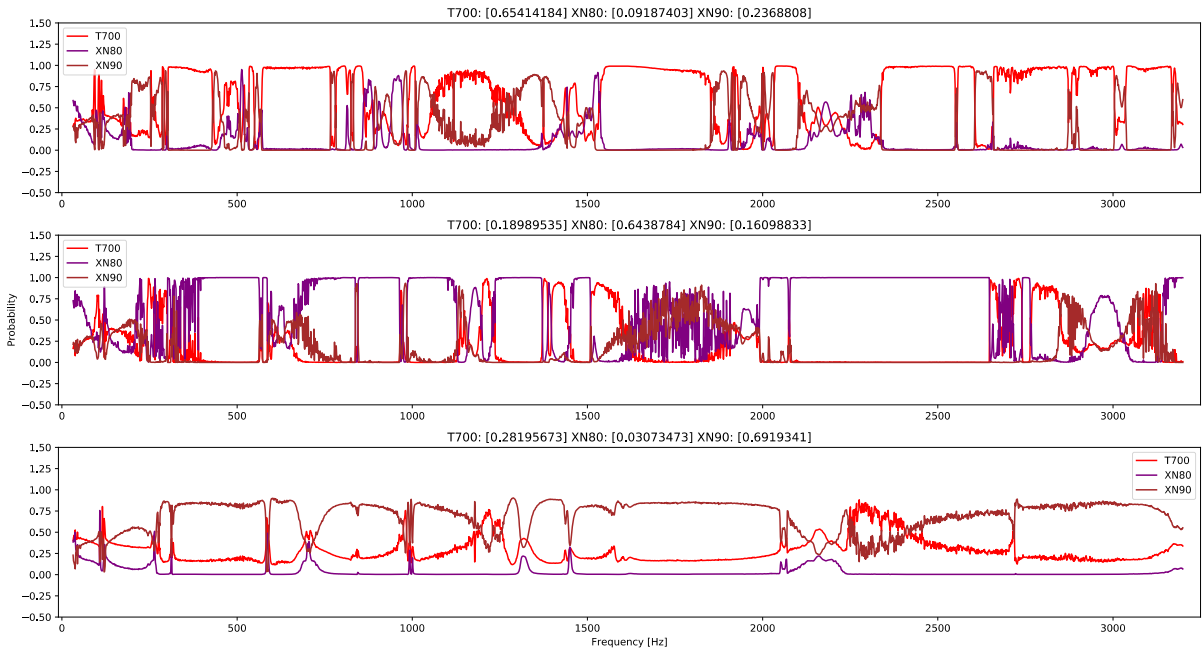
Obrázek 26: Trénovací data klasifikace tří druhů kompozitů pomocí sítě LSTM. Hodnoty pravděpodobností klasifikace jsou vyjádřeny červeně pro T700-N1, fialově pro XN80-N1 a hnědě pro XN90-N1.

Úspěšnosti klasifikace neuronové sítě LSTM pro jednotlivé kompozity vzhledem k frekvenci jsou uvedeny v tabulce 11. Tato úspěšnost opět reprezentuje, kolik procent z celého frekvenčního pásma bylo možné využít k úspěšné klasifikaci.

Tabulka 11: Úspěšnost klasifikace sítě ELM.

T700-N1-780	89,74%
XN80-N1-780	94,16%
XN90-N1-780	95,44%

K otestování takto natrénované sítě jsem použil stejná testovací data jako v případě sítě ELM. Výsledek je uveden na obrázku. Opět jsem využil celé měřené frekvenční pásmo a procentuální pravděpodobnost daného kompozitu jsem vypočetl jako součet příslušného sloupku daného kompozitu podělený celkovým počtem vzorků.



Obrázek 27: Pravděpodobnost klasifikace kompozitů do tří tříd pro celé frekvenční spektrum pomocí sítě LSTM. Červená – T700-N1, fialová – XN80-N1, hnědá XN90-N1.

I tato síť klasifikovala všechna data správně. A to s lepší pravděpodobností než síť ELM. Úspěšnosti neuronové sítě v klasifikaci jednotlivých jednotlivých kompozitů jsou opět uvedeny v tabulce 12.

Tabulka 12: Úspěšnost klasifikace kompozitů dle sítě LSTM na nově skenované sadě

Graf	Kompozit	Pravděpodobnost
1	T700-N1	65,41%
2	XN80-N1	64,38%
3	XN90-N1	69,19%

5.2.3 Aplikace sítě HONU

Síť založenou na jednotkách HONU, která by byla schopná obdobného principu klasifikace se mi bohužel nepovedlo zprovoznit. Počet možných kombinací prvků, které je nutné rozpoznat je opravdu velký a v žádné z mých aplikací jsem nedosáhl požadovaných výsledků. Možnou modifikací této sítě, která by byla schopná dosáhnout požadované klasifikace, je struktura vícevrstvého perceptronu (Multi Layer Perceptron, MPL), kde by jeho neurony by byli typu LNU nebo QNU.

6 Vyhodnocení výsledků

V této kapitole shrnu výsledky dosažené pomocí aplikovaných neuronových sítí ve vztahu k reálným datům. A uvedu zde poznatky, které jsem získal během jejich aplikací a datové přípravě k dosažení uspokojivých výsledků. Celkové porovnání a vhodné aplikace uvedu ale až v samém závěru této práce.

6.1 Data Howden ČKD kompresory

Základní problém byl s velice dlouhou dobou načítání souboru typu excel. Počet dat obsažených v tomto souboru bylo 65 535 řádků v šesti listech s rozdílným množstvím senzorů v každém z nich. Navíc tento soubor obsahoval hodnotu ze senzoru pouze v případě, že nastala dostatečně velké změna, v opačném případě zde byla nastavena hodnota NULL, což bohužel značně ovlivnilo kvalitu poskytnutých dat. Během tohoto zpracování jsem narazil na limitaci objektu DataFrame knihovny Pandas, kterou jsem chtěl využít k nahrazení hodnot null jejich poslední známou hodnotou. Tato knihovna dokáže nahradit určitou hodnotu pouze překopírováním celého objektu se změněnou hodnotou na určitém indexu, což při takto obsáhlých datech trvalo velice dlouho (okolo 30min). Proto jsem místo toho využil převod každého sloupce na objekt numpy array, změny provedl v každém z nich a vytvořil nový objekt typu DataFrame. Po této úpravě byla data doplněna a upravena do jedné minuty.

Jelikož si s úkolem predikce teploty senzoru K12042_TIRSA712B z hodnot senzoru K12042_TIRSA712A poradili všechny sítě velmi obdobně, což bylo dáno i nevhodnými daty, vynechal jsem detailnější hodnocení této aplikace.

6.1.1 Datové předzpracování

Vzhledem ke způsobu zaznamenávání těchto dat jsem jako vhodnou metodu předzpracování zvolil metodu převzorkování. Na hodnotách okolo $n = 30$ pro data z relativních vibrací nebo $n = 60$ pro teplotu. S tímto nastavením dosahovali všechny sítě podstatně nižších hodnot chyby než při jejich nižším počtu. Posledním aplikovaným předzpracováním byla aplikace metody normalizace dat. Po aplikaci metody Z-Score se kvalita predikce všech sítí zhoršila, proto jsem jí v tomto případě vynechal. Predikci jsem ponechal na pouhý jeden vzorek dopředu, jelikož se mi nepodařilo získat uspokojující hodnoty pro predikci více vzorků. Jedná se tedy o velmi krátkodobou predikci.

6.1.2 Aplikace LSTM

Z aplikovaných neuronových sítí dokázala charakteristický průběh relativních vibrací nejlépe vystihnout síť architektury LSTM (obrázek 21). Průběhy predikce této sítě nejlépe odpovídali charakteru průběhu dat , síť ale nedokázala dosáhnout správných peak-to-peak hodnot. Trénování této sítě trvalo také nejdéle ze všech zvolených, a to i přes to, že jako jediná byla schopná využít k trénování všechna procesorová vlákna díky použité knihovně Keras. Vzhledem k širokým možnostem této architektury, bylo obtížné jí zprovoznit, i přes dostupné vzorové aplikace přímo od autorů knihovny, jelikož bylo nutné správně zavést i funkce knihovny TensorFlow, které tato síť využívá ke svému trénování.

6.1.3 Aplikace ELM

Síť ELM měla o něco horší výsledky v charakteristice průběhu, avšak její natrénování bylo velice rychlé a na rozdíl od LSTM dosahuje lepších peak-to-peak hodnot. Ačkoliv tato knihovna má dle svých autorů k dispozici funkci na trénování sítě na více vláknech, i přes veškeré mé pokusy se mi nepovedlo tuto funkci zprovoznit. Pokud by se tak stalo, síť by se natrénovala ještě rychleji než v mé aplikaci. Největší problém této síťové architektury bylo vhodně zvolit velikost skryté vrstvy, která zásadně ovlivňovala přesnost predikce. Dalším problémem ovlivňujícím správnost predikce byly náhodně zvolené váhy, které způsobovaly odlišné kvality výsledků po každém testovacím běhu.

6.1.4 Aplikace HONU

Sítě HONU dosahovali vyšších chyb než dvě dříve zmíněné sítě. Síť LNU nedokázala tuto charakteristiku vystihnout prakticky vůbec a síť QNU, ačkoliv její průběh byl lepší než sítě LNU, také nedosahovala menší chyby než ELM nebo LSTM. Jelikož jsem tyto sítě i objektově programoval, doufal jsem v lepší výsledky alespoň než ELM. Možným zlepšením v tomto ohledu by byla aplikace neuronů kubických neuronů (CNU) nebo sítě typu MLP s neurony HONU, avšak tuto metodiku aplikace nemám v rámci mé bakalářské práce nastudovanou. Největším problémem mé implementace byl správný formát vstupního vektoru do objektu. Proto jsem zavedl funkci reshape_input, která také řešilí i občasnou chybu knihovny numpy, kde objekt array s N prvky a šířce 1 byl někdy navrácen jako $array = (N, 1)$ a někdy jako $array = (N,)$. Jediným správným formátem vstupního vektoru do objektu HONU je $array = (N, n)$, kde n označuje šířku vstupního vektoru.

6.2 Data CompoTech PLUS z vibrometru Polytec

Poskytnutá data byla v binárním souboru navrženém společností Polytec, ke kterému jsem napsal funkci k jejich vyčtení. K tomu mi velmi pomohla příručka objektové architektury Polytec, která se stáhla zároveň se službou Polytec File Access. Díky této příručce jsem mohl pochopit vnitřní strukturu souboru a napsat k tomuto adekvátní funkce v jazyce python. Dalším zjednodušením práce s tímto datovým typem by bylo vyčtení všech možných informací o datech, která jsou v tomto souboru uložena. Bohužel se mi to ale prozatím nepovedlo a bylo tedy nutné přes Polytec File Viewer vždy zjistit, jaká data jsou dostupná v daném souboru.

6.2.1 Datové předzpracování

Jako datovou přípravu jsem opět zvolil převzorkování dat, kde jsem z každých 10 vzorků vždy vybral maximum, čímž charakteristický průběh zůstal zcela zachován, ale povedlo se velmi potlačit hodnoty šumu na určitých frekvencích. Jak je vidět při porovnání obrázků 12 a 23 zvolených 10 vzorků se jeví jako ideální. Při nižších hodnotách byl pouze posílen šum a vyšší hodnoty začínaly tento průběh zkreslovat. Druhé datové předzpracování byla aplikace metody Z-Score, po její aplikaci se neuronovým sítím dařilo klasifikovat dané kompozity s vyšší přesností než pouze po jejich převzorkování. Jako trénovací data jsem si zvolil data z prvního měřeného vzorku a poté náhodně vybral testovací data z měření třetího. Výsledky tohoto přístupu ke klasifikaci se ukázaly jako možné. Pro klasifikaci jsem použil pouze sítě ELM a LSTM, jelikož pro typ HONU by bylo nutné aplikovat architekturu MLP. Důvodem je, že HONU neurony v tomto uspořádání nedosahovali přesnosti ani 10%.

6.2.2 Aplikace ELM a LSTM

Z aplikovaných neuronových sítí dokázala lépe klasifikovat testovací vzorek architektura LSTM, avšak její trénovací časy byly znatelně delší, než tomu bylo u sítě ELM. V případě LSTM se jednalo o 10 minut, zatímco v případě ELM se jednalo o necelých 40 vteřin. Rozdíl těchto trénovacích časů by v případě více trénovacích dat ještě značně narostl. Dalším faktorem, který ovlivňoval přesnost obou neuronových sítí a trénovací čas, byl počet neuronů v každé z nich. V případě LSTM volba více neuronů, než 100 v každé z vrstev měla již velmi malý vliv na přesnost a od počtu 200 neuronů v každé vrstvě se výsledky začali spíše zhoršovat. Celkem tedy tato síť disponovala 300 neurony. Obdobně na tom byla i síť

ELM, maximálním počtem neuronů, které ještě zlepšovali výsledky klasifikace byla hodnota 312, tedy velmi obdobný počet jako v případě LSTM, avšak pouze v jedné vrstvě.

Celkově vzato je tento způsob klasifikace velmi výhodný z možnosti měřit vibrace pouze na relativně úzkém frekvenčním spektru, avšak je potřeba tuto metodiku otestovat na větším množství kompozitů, zda zde dosažené úspěšnosti budou možné i pro ostatní druhy kompozitů.

6.3 Celkové shrnutí výsledků

Predikci časových řad na umělých a reálných datech zvládli do jisté míry všechny sítě, avšak lineární neuronovnou jednotku (LNU) bych nedoporučil k finální aplikaci na žádná z aktuálně testovaných dat, jelikož měla značné problémy během predikce všech komplikovanějších závislostí. Jednotka QNU podávala značně lepší výsledky, čímž se z aplikovaných sítí HONU stala lepší volbou. Sítě typu ELM a LSTM také podávali dobré výsledky. K aplikaci neuronových sítí na predikci dat v závislosti na testovaných datech bych doporučil síť QNU a ELM, obě dvě byli natrénované velice rychle a v závislosti na datech podávali dobré výsledky. Problémem sítě LSTM je vysoká komplexnost celé architektury a s tím spojené velmi dlouhé trénovací časy. Proto bych tuto síť k predikci volil až v případě nedostatečných výsledků sítí QNU a ELM.

Ke klasifikaci kompozitních struktur se mi bohužel nepovedlo zprovoznit sítě typu HONU s dostatečně vysokou přesností, proto jsem klasifikaci pomocí HONU prvního a druhého stupně v mé bakalářské práci vynechal. Řešená úloha evidentně vyžadovala výrazně nelineární klasifikátor. Pomocí sítí typu ELM a LSTM se ovšem klasifikace všech tří testovaných struktur povedla. Z těchto dvou sítí bych ke klasifikaci vybral síť typu LSTM, a to i přes dlouhé trénovací časy. V tomto druhu aplikací je důležitější vysoká spolehlivost výsledku, než dlouhý proces učení. Tento druh sítě poskytl mnohem lepší předpoklady k tomuto účelu aplikace. Síť typu ELM by byla vhodná k rychlému zavedení rozpoznávání nového typu kompozitu, pokud by nebyl dostatečně dlouhý trénovací čas pro síť LSTM.

Závěr

Cílem této bakalářské práce bylo provést rešerši o aktuálním stavu problematiky analýzy průmyslových dat, zjištění druhů používaných neuronových sítích v průmyslu a aplikace čtyř vybraných supervizorovaných neuronových sítí na vybrané obvyklé aplikace v průmyslu.

Hlavním výsledkem rešerše je zjištění, že velkým problémem v průmyslové datové analytice je nesjednocený způsob ukládání a sběru výrobních dat, což značně ztěžuje aplikace jakýchkoliv metod jejich analýzy. Před jakoukoliv aplikací těchto metod, je nutné zařídit unikátní přístup do unikátní databáze nebo souborového systému dané společnosti. Tuto problematiku řeší norma ISO 13374, která vešla v platnost v roce 2012. Tato norma ovšem není závazná a není hojně využívána.

Mezi hlavní cíle analýzy průmyslových dat patří prediktivní údržba strojů a výrobních linek, která má značně snížit náklady celé údržby. Tato prediktivní údržba počítá s využitím neuronových sítí určených k predikci časových řad, detekci poruchových stavů a následné klasifikaci těchto poruchových stavů. Výhodou neuronových sítí je možnost predikce prakticky v reálném čase, jelikož všechny neuronové sítě jsou po naučení velmi rychlé.

V průmyslu nastupují neuronové sítě prozatím velmi pomalu, jelikož je velmi důležitá stabilita a spolehlivost všech použitých metod z důvodů vysokých finančních ztrát v případě nečekaného odstavení výroby. Aktuálně se jeví supervizorované metody jako výhodnější, jelikož na rozdíl od nesupervizorovaných sítí je možné zjistit na základě čeho takovéto sítě rozhodují.

Z neuronových sítí jsem si zvolil architektury LSTM, ELM a HONU. Ze skupiny HONU jsem si zvolil sítě LNU a QNU. Jako obvyklé aplikace jsem si vzhledem k dostupným reálným datům vybral predikci časové řady a predikci hodnot jednoho senzoru z hodnot senzoru druhého pro data z kompresoru společnosti HČKD. A vzhledem k charakteru druhých reálných dat od společnosti CompoTech PLUS, jsem si zvolil klasifikaci, která je obvykle součástí detekce poruchových stavů.

V Pythonu jsem naprogramoval skripty, které provedou automatickou aplikaci predikce nebo klasifikace pomocí sítí LSTM, ELM a HONU. Pro mnou naprogramované HONU jsem použil učící algoritmy Levenberg-Marquardt a Gradient Descent. Pro data použitá v této práci jsem nastavil všechny potřebné parametry a napsal funkce, které po jejich spuštění provedou

všechny požadované operace ke spuštění učícího procesu a zobrazení patřičných výsledných hodnot.

Knihovna aplikovaných neuronových sítí je navržena tak, aby bylo jednoduché jí dle potřeby modifikovat. Je tedy možné jí do budoucna rozšiřovat a to jak z hlediska nových algoritmů učení, tak i neuronových sítí. V navazujících pracích by bylo vhodné provést učení k predikci na kvalitnějších reálných datech s jejich vhodným předzpracováním a v případě klasifikace by bylo vhodné lépe připravit data a vzít v úvahu hodnotu odstupu signál-šum, což nebylo předmětem této práce. Na takto lépe připravených datech poté natrénovat komplexnější neuronovou síť, která by byla schopná rozpoznat všechny druhy daných kompozitů.

V práci jsem se snažil shrnout dostupné teoretické poznatky o aktuálním stavu datové analytiky v průmyslu pomocí supervizorovaných neuronovaných sítí. Dále jsem popsal a implementoval zadané základní neuronové sítě včetně pokročilých sítí typu LSTM a aplikoval na konkrétní reálná data. Poznatky z mé práce mohou posloužit k dalšímu vývoji usnadnění aplikace na jiná reálná data a jejich systémy.

Zdroje

[1] KRUPA, Miroslav. Prediktivní údržba a metody technické prognostiky – seznámení se s problematikou [online]. 2011, **4**(4). ISSN 1803-3687. Dostupné z: https://www.bozpinfo.cz/josra/prediktivni-udrzba-metody-technicke-prognostiky-seznameni-se-s-problematikou

[2] SUNG, H. J. *Optimal maintenance of a multi-unit system under dependencies : a thesis presented to the academic faculty* [online]. Georgia, 2008 [vid. 2019-03-06]. Georgia Institute of Technology. Dostupné z: https://smartech.gatech.edu/handle/1853/26511

[3] PALADE, Vasile, Cosmin Danut BOCANIALA a L. C. JAIN, ed. *Computational intelligence in fault diagnosis*. London: Springer, 2006. Advanced information and knowledge processing. ISBN 978-1-84628-343-7.

[4] LUO, Jianhui, Madhavi NAMBURU, Krishna PATTIPATI, Liu QIAO, Masayuki KAWAMOTO a Shunsuke CHIGUSA. Model-based Prognostic Techniques [online]. nedatováno [vid. 2019-03-11]. Dostupné z: https://www.teamqsi.com/wp-content/uploads/2012/10/063Luo.pdf

[5] YANG, C., W. SHEN a X. WANG. The Internet of Things in Manufacturing: Key Issues and Potential Applications. *IEEE Systems, Man, and Cybernetics Magazine* [online]. 2018, **4**(1), 6–15. ISSN 2333-942X. Dostupné z: doi:10.1109/MSMC.2017.2702391

[6] *Analýzy vad při zabezpečování spolehlivosti elektronických součástek* [online]. [vid. 2019-03-13]. Dostupné z: http://www.elektrorevue.cz/clanky/01028/index.html

[7] FIBIR, Tomáš. *Detekce poruchových stavů dynamických systémů* [online]. Praha, 2003 [vid. 2019-03-14]. Diplomová práce. ČVUT Fakulta elektrotechnická. Dostupné z: https://support.dce.felk.cvut.cz/mediawiki/images/9/9d/Dp_2003_fibir_tomas.pdf

[8] Machine fault detection and classification by pattern recognition. *RSIP Vision* [online]. 17. listopad 2016 [vid. 2019-03-21]. Dostupné z: https://www.rsipvision.com/machine-fault-detection-and-classification/

[9] POKORNÝ, Jan. *Aplikace výpočetní inteligence v řešení bezpečnosti silničního provozu* [online]. Pardubice, 2010 [vid. 2019-03-22]. Disertační Práce. Pardubice, Dopravní Fakulta Jana Pernera. Dostupné z: https://dk.upce.cz/bitstream/handle/10195/42152/PokornyJ_AplikaceVypocetni_MT_2011.pdf?sequence=4&isAllowed=y

[10] BÍLA, ŠMÍN, KRÁL a HLAVÁČ. *Informační technologie. Databázové a znalostní systémy.* B.m.: ČVUT, 2009.

[11] HERRERA, Francisco, Alberto FERNÁNDEZ a Isaac TRIGUERO. Fuzzy Models for Data Science and Big Data. In: *IEEE International Conference on Fuzzy Systems* [online]. Konference. Naples, Italy. 12.6 2017 [vid. 2019-03-20]. Dostupné z: http://www.cs.nott.ac.uk/~pszit/download/Tutorial-FuzzyBigData.pdf

[12] *Úvod do problematiky umělých neuronových sítí* [online]. [vid. 2019-03-22]. Dostupné z: http://www.elektrorevue.cz/clanky/00013/index.html

[13] GUPTA, M Madan, Ivo BUKOVSKÝ, M. G. Ashu SOLO, N. HOMMA a Z-G. HOU. Fundamentals of Higher Order Neural Networks for Modeling and Simulation. In: *Artificial Higher Order Neural Networks for Modeling and Simulation* [online]. B.m.: IGI Global, 2012 [vid. 2019-03-25], s. 103–133. ISBN 978-1-4666-2175-6. Dostupné z: https://www.researchgate.net/publication/215884506_Fundamentals_of_Higher_Order_Neural_Networks_for_Modeling_and_Simulation

[14] ZMEŠKAL, Jiří. *Extrémní učící se stroje pro předpovídání časových řad* [online]. Brno, 2018 [vid. 2019-03-25]. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=172598

[15] LUNER, Petr. *Genetické algoritmy* [online]. [vid. 2019-03-25]. Dostupné z: https://cgg.mff.cuni.cz/~pepca/prg022/luner.html

[16] TEDA, Jaroslav. Genetické algoritmy a jejich aplikace v praxi. *Programujte.com* [online]. [vid. 2019-03-25]. Dostupné z: http://programujte.com/clanek/2005072601-geneticke-algoritmy-a-jejich-aplikace-v-praxi/

[17] ODUGUWA, V., A. TIWARI a R. ROY. Evolutionary computing in manufacturing industry: an overview of recent applications. *Applied Soft Computing* [online]. 2005, **5**(3), Application Reviews, 281–299. ISSN 1568-4946. Dostupné z: doi:10.1016/j.asoc.2004.08.003

[18] *Artificial Neural Network | Quantra by QuantInsti* [online]. [vid. 2019-03-26]. Dostupné z: https://quantra.quantinsti.com/glossary/Artificial-Neural-Network

[19] Okuma presents new AI-infused CNC control. *Mould&Die World magazine* [online]. 23. prosinec 2016 [vid. 2019-03-26]. Dostupné z: http://www.mouldanddieworld.com/okuma-presents-new-cnc-control/

[20] SCHMID, Martin a MRÁZOVÁ. *Konvoluční neuronové sítě a jejich implementace* [online]. Praha, 2011 [vid. 2019-03-27]. Bakalářská práce. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta. Dostupné z: https://is.cuni.cz/webapps/zzp/download/130041723/?lang=cs

[21] BUKOVSKY, Ivo a Noriyasu HOMMA. An Approach to Stable Gradient-Descent Adaptation of Higher Order Neural Units. *IEEE Transactions on Neural Networks and Learning Systems* [online]. 2016, 1–13. ISSN 2162-237X, 2162-2388. Dostupné z: doi:10.1109/TNNLS.2016.2572310

[22] *Extreme learning machine: Theory and applications - ScienceDirect* [online]. [vid. 2019-04-11]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0925231206000385

[23] HUANG, Guang-Bin a Lei CHEN. Convex incremental extreme learning machine. *Neurocomputing* [online]. 2007, **70**(16–18), 3056–3062. ISSN 09252312. Dostupné z: doi:10.1016/j.neucom.2007.02.009

[24] GUO, Jiang. BackPropagation Through Time. In: [online]. B.m.: SemanticScholar, 2013, s. 6. Dostupné z: https://www.semanticscholar.org/paper/BackPropagation-Through-Time-Guo/c77f7264096cc9555cd0533c0dc28e909f9977f2

[25] HOCHREITER, Sepp a Jürgen SCHMIDHUBER. Long Short-term Memory. *Neural computation* [online]. 1997, **9**, 1735–80. Dostupné z: doi:10.1162/neco.1997.9.8.1735

[26] GERS, F. A. a J. SCHMIDHUBER. Recurrent nets that time and count. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium* [online]. 2000, s. 189–194 roč.3. Dostupné z: doi:10.1109/IJCNN.2000.861302

[27] *Tips for Training Recurrent Neural Networks* [online]. [vid. 2019-04-16]. Dostupné z: https://danijar.com/tips-for-training-recurrent-neural-networks/

[28] SCHRAUDOLPH, Nic a Fred CUMMINS. Linear Neural Networks. *Introduction to Neural Networks* [online]. 2006 [vid. 2019-03-29]. Dostupné z: https://cnl.salk.edu/~schraudo/teach/NNcourse/linear2.html

[29] MORÉ, Jorge J. The Levenberg-Marquardt algorithm: Implementation and theory. In: G. A. WATSON, ed. *Numerical Analysis* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978 [vid. 2019-05-14], s. 105–116. ISBN 978-3-540-08538-6. Dostupné z: doi:10.1007/BFb0067700

[30] *Hessian matrix of scalar function - MATLAB hessian* [online]. [vid. 2019-05-14]. Dostupné z: https://www.mathworks.com/help/symbolic/hessian.html#buiej1q-2

[31] LEVENBERG, KENNETH. A METHOD FOR THE SOLUTION OF CERTAIN NON-LINEAR PROBLEMS IN LEAST SQUARES. *Quarterly of Applied Mathematics*. 1944, **2**(2), 164–168. ISSN 0033-569X.

[32] *File Format Specification* [online]. [vid. 2019-04-26]. Dostupné z: https://portal.hdfgroup.org/display/HDF5/File+Format+Specification

[33] AKUSOK, A., K. BJÖRK, Y. MICHE a A. LENDASSE. High-Performance Extreme Learning Machines: A Complete Toolbox for Big Data Applications. *IEEE Access* [online]. 2015, **3**, 1011–1025. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2015.2450498

[34] *Home - Keras Documentation* [online]. [vid. 2019-05-23]. Dostupné z: https://keras.io/

[35] A Beginner's Guide to LSTMs and Recurrent Neural Networks. *Skymind* [online]. [vid. 2019-04-11]. Dostupné z: http://skymind.ai/wiki/lstm

7 Příloha

Seznam přílohy:

1. Instrukce k instalaci prostředí pro spuštění mých aplikací
2. Instrukce k zobrazení vzorových aplikací
3. Zdrojový kód
4. CD s prací v elektronické podobě, funkcemi a obrázky

7.1 Instalace prostředí

Postup instalace:

1. Instalace Pythonu 3.7 (testováno na 3.7.3) pomocí služby Anaconda z https://www.anaconda.com/
   - Nainstalovat instalační balíček pro vhodný OS dle jeho instrukcí
   - aktualizace Python nástrojů příkazem v Anaconda Prompt:
   - **conda update --all**
2. Instalace nezbytných balíčků příkazem v Anaconda prompt v následujícím pořadí:
   - **conda install tensorflow sklearn keras d6tstack**
   - **python -m pip install hpelm**
3. Instalace služby Polytec File Access pomocí Polytec Update z https://www.polytec.com/eu/vibrometry/products/software/polytec-update/
   - Nainstalovat Polytec Update pro vhodný OS dle jeho instrukcí
   - Po spuštění tohoto softwaru vybrat vše a spustit automatickou instalaci.
   - Po instalaci je nutné restartovat PC

7.2 Instrukce k zobrazení vzorových aplikací

V případě požadavku na jiná data, je nutné je vložit do příslušné podsložky v „pouzita_data“. Použité složky jsou náležitě pojmenovány. Ve složce „HCKD kompresory“ je očekáván vstupní excel, případně náležitě upravené csv soubory včetně konfiguračního txt. V případě dat z vibrometru Polytec se očekává jako vstupní soubor svd. Alternativně je možné změnit přístupové cesty přímo v hlavním skriptu „main.py“

K zobrazení zde užitých grafů slouží funkce ve skriptech „main.py“, „polytec_svd_read.py“ a „HCKD_data_preprocess.py“. Funkce zobrazení grafu jsou vždy připravené na konci každého skriptu a lze je aktivovat odkomentováním požadované spouštějící funkce. Kromě grafů jsou tímto způsobem připraveny i ostatní funkce.

7.3 Zdrojové kódy

V mé bakalářské práci užívám 5 skriptů.

Seznam skriptů:

1. Hlavní skript
    - main.py – propojení použitých skriptů a implementace zmíněných aplikací v mé bakalářské práci. **Na konci skriptu jsou připravené příkazy k demonstraci užitých implementacích, které se spustí odkomentováním jedné z funkcí sample_x( ).**
2. Přídavné moduly
    1. neural_units.py – modul ve kterém jsou objektově naprogramované neuronové sítě HONU, konkrétně LNU a QNU a jejich učící algoritmy metody zpětného šíření GD a LM.
    2. polytec_svd_read.py – modul sloužící k přístupu do datových souborů typu svd používaných společností Polytec. Přístup je umožněn pomocí objektové komunikace standardem windows COM nebo jeho ekvivalentem na jiných OS. **Na konci skriptu jsou připravené příkazy k demonstraci vyčtení dat ze souboru a jejich zobrazením do grafu.**

3. Dodatečné skripty
    1. umela_data.py – skript sloužící k vytvoření umělých dat použitých v této bakalářské práci. Po spuštění se vygenerovaná data uloží do souborového formátu csv.
    2. HCKD_data_preprocess.py – skript sloužící k převodu vstupního souboru excel až po závislostní bloky uvedené v config.txt. **Skript se spouští postupným odkomentováním funkcí na konci souboru. Je zde také připraven skript k vykreslení grafu teplot.**

Okomentování kódu

- Skripty main.py a neural_units.py jsou okomentovány důkladně, jelikož v nich probíhá implementace neuronových sítí.
- Skripty HCKD_data_preprocess.py, polytec_svd_read.py a umela_data.py jsou okomentovány pouze do té míry, aby bylo jasné jejich chování.