VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

RESERVOIR COMPUTING PRO PRŮMYSLOVÉ APLIKACE

RESERVOIR COMPUTING FOR INDUSTRIAL APPLICATIONS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Jakub Brhel

AUTHOR

VEDOUCÍ PRÁCE — Ing. Jiří Kovář, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Jakub Brhel**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Jiří Kovář, Ph.D.**
Akademický rok:	2022/23

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Reservoir Computing pro průmyslové aplikace

Stručná charakteristika problematiky úkolu:

Reservoir computing je výpočetní rámec odvozený z teorie rekurentních neuronových sítí, který mapuje vstupní signály do výpočetních prostorů vyšších dimenzí prostřednictvím dynamiky pevného nelineárního systému nazývaného reservoir. Cílem práce je prozkoumat a použít reservoir computing pro část obráběcího stroje.

Cíle bakalářské práce:

1) Proveďte rešerši metod v oblasti tzv. reservoir computingu a zhodnoťte jednotlivá řešení s uvážením vlastností sledované technické soustavy
2) Navrhněte softwarové řešení na základě provedené rešerše
3) Proveďte test vytvořeného softwarového řešení na reálných datech získaných z měření sledované technické soustavy

Seznam doporučené literatury:

NAKAJIMA, K., FISCHER, I. Reservoir Computing - Theory, Physical Implementations, and Applications, 2021,ISBN 978-981-13-1687-6

GERON, A., Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2019, ISBN-13: 978-1492032649

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2022/23

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

V této bakalářské práci je zkoumána oblast reservoir computingu a jeho aplikace v průmyslovém odvětví. Hlavním cílem práce je ověřit účinnost a přesnost metody echo state network v predikci časových řad. K dosažení tohoto cíle je použit dataset z výrobního stroje a je implementován algoritmus echo state network. Jsou analyzovány výsledky a jsou rovněž porovnány s výsledky předchozích studií v oblasti reservoir computingu.

ABSTRACT

In this bachelor thesis, the field of reservoir computing and its application in the industrial sector is investigated. The main objective of the thesis is to verify the effectiveness and accuracy of the echo state network method in time series prediction. To achieve this objective, a dataset from a manufacturing machine is used and the echo state network algorithm is implemented. The results are analysed and are also compared with the results of previous studies in the field of reservoir computing.

KLÍČOVÁ SLOVA

reservoir computing, aplikace reservoir computingu, strojové učení, predikce časových řad, rekurentní neuronové sítě, hluboké učení

KEYWORDS

reservoir computing, application of reservoir computing, machine learning, time series prediction, recurrent neural networks, deep learning

BIBLIOGRAFICKÁ CITACE

BRHEL, Jakub. *Reservoir Computing pro průmyslové aplikace.* Brno, 2023. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/145597. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Jiří Kovář.

PROHLÁŠENÍ

Prohlašuji, že jsem *bakalářskou* práci na téma **Reservoir Computing pro průmyslové aplikace** vypracoval samostatně s použitím odborné literatury a pramenů, uvedených v seznamu, který tvoří přílohu této práce

Datum

Jakub Brhel

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Jiřímu Kováři, Ph.D. za ochotu, vstřícnost, odborné vedení, trpělivost a rady při zpracovávání mé bakalářské práce

OBSAH

1. Úvod

Neuronové sítě se v posledních letech staly populárním nástrojem pro řešení komplexních matematických problémů, pro předpovědi fyzikálních dějů, pro tvorbu simulací a umělé inteligence. Problematika vytvoření dané neuronové sítě je úzce svázána s množstvím dat, se kterým můžeme tuto síť trénovat, a samotnou velikostí této sítě. Vznikla potřeba pro navržení nových modelů neuronových sítí, které vyžadují menší množství dat na trénink a nejsou náročnější na výpočet. Jedna z metod, která se zabývá tímto problémem, je právě reservoir computing.

Reservoir computing je metoda umělé inteligence, která se zaměřuje na zjednodušení vývoje neuronových sítí. Cílem reservoir computingu je zmenšení počtu trénovatelných neuronových vazeb v neuronové síti. Tím by došlo k značnému urychlení tvorby funkčních neuronových sítí. Neuronová síť se skládá ze vstupní, vnitřní (též "reservoir") a výstupní vrstvy. Reservoir je tvořen množstvím neuronů, které jsou mezi sebou propojeny řídkými a náhodnými vazbami. Tyto vazby jsou před trénováním sítě inicializovány náhodnými hodnotami a poté jsou pevně dané, tedy netrénují se. Vstupní signál je poté přiveden do sítě a prochází několika náhodně vybranými neurony v reservoiru, kde dochází k jeho transformaci. Tento transformovaný signál je poté přiveden do vrstvy výstupní, kde proběhne samotná klasifikace nebo predikce. Touto metodou je docíleno toho, že neuronová síť nevyžaduje trénování vnitřních vazeb, což výrazně zmenšuje množství dat potřebných pro trénování celé sítě.

Motivací pro tuto práci je predikce přesnosti výrobní buňky. Tato přesnost je závislá na teplotě, při které stroj operuje, a je proto příhodné předpovídat přesnost za použití umělé inteligence. Tato práce je zaměřena na vytvoření predikčního modelu, který bude schopen předpovídat přesnost výrobní buňky na základě teploty, při které stroj pracuje. Pro trénování modelu jsou použity historická data z výrobního procesu. Tento model by mohl být v budoucnu použit v průmyslovém sektoru pro zlepšení kvality výroby a optimalizaci výrobních procesů.

2 Rešerše reservoir computing

Reservoir computing má v dnešní době široké spektrum využití v oblastech, jako jsou rozpoznávání řeči, analýza obrazu, analýza časových řad a další. V závislosti na velikosti tréninkových dat a množství parametrů je nutné vybrat účinnou tréninkovou metodu, jinak může docházet k pomalému, nebo neúspěšnému tréninku neuronových sítí. V této rešerši jsou vysvětleny základy trénování neuronových sítí, jakými metodami a způsoby je možné provádět trénink neuronových sítí, jejich výhody a nevýhody. Dále se rešerše zaobírá rozdíly v metodách reservoir computingu a výhodami reservoir computingu oproti běžným neuronovým sítím.

2.1 Střední kvadratická chyba

Střední kvadratická chyba (MSE) je veličina reprezentující rozdíl mezi naměřenou veličinou a reálnou hodnotou. Měření nejsou nikdy přesná a neshodují se s reálnými hodnotami, z toho důvodu je tato hodnota výhradně kladná a nenulová. Střední kvadratická chyba je stavebním kamenem mnoha následujících algoritmů, zejména pro nalezení minima ztrátové funkce.

$$\mathrm{MSE}(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 \qquad (2.1)$$

θ je vektor parametrů minimalizující ztrátovou funkci

$x^{(i)}$ je vstupní vektor hodnot z datasetu v kroku i

$y^{(i)}$ je výstupní vektor kroku i

m je počet prvků v datasetu

Z rovnice vidíme, že je využíváno naměřených hodnot společně s parametrem minimalizující ztrátovou funkci. Právě nalezení tohoto parametru je zásadní pro vytvoření spolehlivé a kvalitní neuronové sítě.

2.2 Normálová rovnice pro lineární regresi

Jedná se o analytickou metodu schopnou jednoznačného určení minima ztrátové funkce pomocí MSE.

Máme hypotetickou funkci:

$$h(\theta) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T x \qquad (2.2)$$

kde θ_0 je bias funkce a $x_0 = 1$

Cílem je nalézt minimum ztrátové funkce.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2m}(X\theta - Y)^T(X\theta - Y) \tag{2.3}$$

$$\frac{\delta J(\theta)}{\delta\theta} = 2X^TX\theta - 2X^TY = 0 \tag{2.4}$$

$$\theta = (X^TX)^{-1}X^TY \tag{2.5}$$

θ je vektor parametrů minimalizující ztrátovou funkci

m je počet prvků v datasetu

X je vektor všech vstupních hodnot do neuronové sítě

Y je vektor všech výsledných hodnot z neuronové sítě

Pro tuto metodu však představuje zásadní problém právě potřeba celého datasetu s tréninkovými daty. Jak již bylo zmíněno, pro trénink neuronových sítí je potřeba velké množství dat. Pokud bychom se pokusili invertovat $(X^TX)^{-1}$, museli bychom invertovat matici o velikosti n × n. Vzhledem k tomu, že čas k přetvoření matice nabývá kubicky $O(n^3)$, dříve by došlo k vyčerpání operační paměti než k vypočtení této rovnice [1]. Mohou nastat případy, kdy pracujeme s malými datasety, ale tato metoda je pro naše účely nežádoucí. Je potřeba použít metod, které nevyužívají celý dataset a jsou schopny nalézt minimum ztrátové funkce.

2.3 Gradientní sestup

Obecný zápis vypadá následovně:

$$\theta_{n+1} = \theta_n - \eta\nabla\mathrm{MSE}(\theta_n) \tag{2.6}$$

Kde:

θ_n je vektor parametrů v kroku n

θ_{n+1} je vektor parametrů v kroku $n + 1$

η je learning rate

Kde $\nabla\mathrm{MSE}(\theta_n)$ je derivací MSE podle každého prvku parametru θ. Tedy $\frac{\delta}{\delta\theta_j}\mathrm{MSE}(\theta) = \frac{2}{m}\sum_{i=1}^{m}(\theta^Tx^{(i)} - y^{(i)})x_j^{(i)}$.

Využíváme toho, že optimální parametry vah jsou minimem funkce, kterou je možno nalézt, pokud v každém kroku budeme postupovat proti stoupání této funkce (proto $-\eta\nabla F(\theta_n)$). Rychlost učení (learning rate) je důležitou součástí této rovnice, protože

udává velikost kroku, jímž dané minimum najdeme. Příliš velká velikost kroku může způsobit divergenci a minimum funkce by nebylo možné nalézt. Naopak příliš malá velikost kroku znamená, že minimum by bylo hledáno zbytečně dlouho, nicméně bylo by nalezeno [1].

2.3.1 Batch Gradient Descent

Batch Gradient Descent je jednou z nejintuitivnějších metod. Zhodnocením vstupních dat můžeme určit derivaci, po jejímž záporném směru je možné se posunout. Touto metodou je možné nalézt nejpřímější cestu k minimu funkce.

$$\theta_{n+1} = \theta_n - \eta \nabla \mathrm{MSE}(\theta_n) \tag{2.7}$$

$$\nabla \mathrm{MSE}(\theta_n) = \frac{2}{m} X^T (X\,\theta_n - Y) \tag{2.8}$$

$$\theta_{n+1} = \theta_n - \eta \frac{2}{m} X^T (X\,\theta_n - Y) \tag{2.9}$$

Přestože se znovu objevuje celý dataset, již není invertovaný a je tedy lepší alternativou pro trénink velkých datasetů než normálová rovnice pro lineární regresi [1].

2.3.2 Stochastic Gradient Descent

Některé data sety mohou nabývat takových rozměrů, až nemůžeme použít celý data set. Zdánlivě jednoduchým řešením je vždy vzít pouze jednu náhodnou hodnotu datasetu a z této hodnoty vypočítat gradient. Jednou z nevýhod této metody je nestabilní konvergence k minimu ztrátové funkce, jež způsobuje, že naprostého minima nemůže být dosaženo, ale budeme se pohybovat v jeho blízkosti [1].

2.3.3 Mini-batch Gradient Descent

Tato metoda je vhodným kompromisem mezi Stochastic Gradient Descent a Batch Gradient Descent. Pokud stále pracujeme s příliš velkým datasetem, nicméně chceme přesnější hodnotu minima. Jednoduchým řešením je vzít více náhodných hodnot datasetu a vypočítat průměrný gradient z jednotlivých gradientů těchto náhodných hodnot. Konvergence k výsledku stále nebude stabilní, avšak bude přesnější než Stochastic Gradient Descent. Další výhodou je využití vektorové optimalizace, která urychluje výpočetní čas [1].

2.3.4 Algoritmus zpětného šíření

Jedná se o algoritmus vypočítávající gradient ztrátové funkce pro jednotlivé váhy synapsí. Jak název napovídá, tento algoritmus počítá rozdíl hodnot, které vyšly z neuronové sítě, a požadovaných hodnot, následně počítá, jak moc jednotlivé váhy v neuronové síti k tomuto rozdílu přispěly. Tento gradient je používán v gradientním sestupu. Nicméně čím větší neuronová síť je, tím více se gradient v hlubších vrstvách zmenšuje, až následně nedochází ke skoro žádným změnám v rámci vah. Tento problém se nazývá mizející gradient a z velké části se řeší použitím praktičtějších aktivačních funkcí. Na druhou stranu u rekurentních neuronových sítí dochází k pravému opaku, kdy se kumulují

gradienty, dokud nedojde k divergenci celé soustavy. Tomuto se dá zabránit omezením velikosti gradientu, nebo samotných vah.

2.4 Regularizované lineární modely

Velké množství stupňů volnosti, jež je způsobeno velkým množstvím výstupních vah, může způsobovat overfitting, tzn. je vytvořen zbytečně složitý model, který není vhodný pro ostatní datasety. Řešením takového problému je regularizace modelu, která tlumí váhy s přebytečnými informacemi a tím odebírá stupně volnosti.

2.4.1 Lasso regularizace

K ztrátové funkci přičteme penalizační prvek $L1 = \alpha \sum_{i=1}^{n} |\theta_i|$

$$J(\theta) = \mathrm{MSE}(\theta) + \alpha \sum_{i=1}^{n} |\theta_i| \tag{2.10}$$

Jelikož se v prvku objevuje absolutní hodnota, nemá funkce jednoznačné řešení. Řešení je tedy pouze přibližné a jeho výpočet trvá déle. Nicméně touto metodou jsme schopni určit, které parametry jsou pro nás nepodstatné, a zmenšit je na hodnoty blížící se nule.

2.4.2 Ridge Regression

K ztrátové funkci přičteme penalizační prvek $L2 = \frac{1}{2}\alpha \sum_{i=1}^{n} \theta_i^2$

$$J(\theta) = \mathrm{MSE}(\theta) + \frac{1}{2}\alpha \sum_{i=1}^{n} \theta_i^2 \tag{2.11}$$

$$\theta = (X^T X + \alpha A)^{-1} X^T Y \tag{2.12}$$

Pomocí parametru α se zesiluje nebo zeslabuje síla regularizace. Pokud $\alpha = 0$, jedná se o obyčejnou lineární regresi. Pokud by však α bylo nastaveno na příliš vysokou hodnotu, velikost parametru θ bude příliš malá a došlo by naopak k underfittingu. Správným použitím této metody regulujeme velké množství parametrů, a je tedy užitečným nástrojem proti overfittingu [1].

2.4.3 Elastic net

Pokud je třeba využít obou prvků $L1$ a $L2$, využívá se elastic net. Elastic net je kombinací obou metod s přidaným parametrem r pro určení silového poměru mezi prvkem $L1$ a $L2$, kde $r = 0$ omezuje prvek $L1$ a $r = 1$ omezuje prvek $L2$.

$$J(\theta) = \mathrm{MSE}(\theta) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^{n} \theta_i^2 \tag{2.13}$$

2.5 Overfitting/Underfitting

Při tréninku neuronových sítí je zaznamenáván i šum, který se v datasetu běžně vyskytuje, což způsobí, že ztrátová funkce tréninkového datasetu je téměř nulová, jelikož neuronové sítě zahrnují tyto chyby, nicméně ztrátová funkce testovacího datasetu začne nabývat vyšších hodnot. Je tedy vytvořena neuronová síť, u níž není možné zpracovávat nová data. Přetrénování (overfitting) je častý problém u malých datasetů, právě protože se šum jednoduše projevuje. Underfitting je naopak způsoben značným omezením stupňů volnosti neuronových sítí, což se projevuje vysokými hodnotami jak u tréninkových, tak testovacích dat ztrátových funkcí.

2.5.1 Cross Validation

Rozdělením tréninkových dat na tréninková, tedy data, kterými jsou vytvářeny vazby v neuronových sítích, a testovací, tedy data, které se nepodílí na tvorbě vazeb v neuronových sítích, ověříme, zda v síti nedochází k overfittingu či underfittingu.

2.5.2 K-fold Cross Validation

Jedním ze způsobů pro cross validation je rozdělit dataset do k složek (folds). Je vybrána jedna složka, která slouží k testování, zbývajících $k-1$ složek slouží k tréninku neuronové sítě. Tento proces je opakován, dokud nejsou všechny složky použity k testování neuronové sítě. Výsledkem je průměr z jednotlivých výsledků.

2.5.3 Leave One Out Cross Validation (LOOCV)

Další možností cross validation je LOOCV. Pokud $k = n$, kde n je počet dat v datasetu, vzniká forma k-fold validation, v níž je použit celý dataset pro trénink a pouze jeden řádek dat je použit na test. Tato metoda má však své nedostatky. Velká hodnota k prodlužuje procesní čas a použitím omezeného množství dat pro testování je snížena variace, se kterou můžeme zhodnotit síť.

2.6 Typy systémů strojového učení

Typ tréninku se liší v závislosti na úkolu, který chceme, aby neuronové sítě vykonávaly. Přestože se pro například předpověď teploty a rozpoznání obličeje na fotce používají neuronové sítě, obě tyto sítě musí z podstaty používat jiný systém pro získávání dat a pro jejich následný trénink.

2.6.1 Učení s učitelem/bez učitele

Prvním rozdílem těchto dvou typů učení je, zda jsou známy kategorie, do kterých data spadají. Pokud dokážeme kategorizovat data, provádí se učení s učitelem. Pokud nevíme, do jaké kategorie data spadají, a chceme nalézt anomálie v těchto datech, využívá se učení bez učitele.

Učení s učitelem

Učení s učitelem je metoda, při níž do neuronových sítí vkládáme data, o kterých již víme, do jaké kategorie patří. Tyto data se označují tzv. labelem, doslovně štítkem, který data kategorizuje. Příkladem tréninku takové neuronové sítě může být právě rozpoznání obličeje. Při tréninku se do neuronové sítě vkládají obrázky s labely, které udávají, jestli

obsahují nebo neobsahují obličej. Po tréninku daná neuronová síť dokáže s určitou pravděpodobností určit, jestli je nebo není na obrázku obličej.

Učení bez učitele

Učení bez učitele pracuje na zcela opačném principu. Do neuronové sítě se vkládají data, o kterých nic není známo a je tedy potřeba je kategorizovat. Příkladem je sledování návštěvníků internetových stránek. O těchto lidech jsou sbírány data jako pohlaví, čas navštívení a podobně. Tyto informace se vloží do neuronové sítě, aby se našel určitý vzorec chování u těchto dat. Tento systém může mít následně využití pro cílené reklamy na webových stránkách dle získaných dat o uživatelích.

Kombinace učení s učitelem a bez učitele

Kombinace učení s učitelem a bez učitele, jak název vypovídá, využívá částečně obou metod. Pro příklad se vraťme k rozpoznávání obličeje. Po vložení obrázků a fotek neuronová síť rozpoznává na fotkách obličej, který patří jednomu člověku. Jediné, co je třeba, je uvést u jedné z fotek, o jakou osobu se jedná, a neuronová síť už s určitou pravděpodobností dokáže kategorizovat obrázky s touto osobou.

2.6.2 Učení online/offline

Dalším kritériem pro řazení typů tréninků neuronových sítí je, zda je možné vytrénovat neuronové sítě a využívat je bez aktualizací, a nebo je nutné do neuronových sítí vkládat stále nová data a aktualizovat je pro adaptaci na momentální situaci.

Batch learning

Batch learning je metoda tréninku neuronových sítí, v níž je neuronová síť trénována celým momentálně dostupným datasetem. Pokud je nutné neuronovou síť aktualizovat o nová data, je nutné použít celý předchozí dataset s novými daty a trénovat celou neuronovou síť od začátku. Přestože je možné proces aktualizace automatizovat, je potřeba velké množství času a procesní paměti.

Učení online

Data se do neuronové sítě neposílají jako celý dataset, ale buď individuálně nebo po malých částech (mini batches). Takto může neuronová síť přijímat data jako nepřetržitý proud informací, kterými může být neustále aktualizována. Pokud se objeví nová data, je možné je rovnou poslat do neuronové sítě. Pokud jsou tato data zcela nového charakteru, začne se neuronová síť aktualizovat změnou vah tak, aby nová data odpovídala očekávanému výsledku. Tyto data však mohou být pouze anomálií a je tedy nutné zvolit přiměřenou rychlost učení. Vzhledem k možnosti neustálé aktualizace neuronové sítě je tato metoda vhodnou alternativou pro batch learning v případě, kdy již není možné trénovat neuronovou síť s pomocí celého datasetu. Protože by nebylo možné dataset načíst jako celek, využívá se právě učení online, v tomto případě Out-of-core learning, kdy je dataset posílán po částech.

2.6.3 Instance-based learning

Instance-based learning je forma kategorizace nových testovacích dat v závislosti na podobnosti s dříve získanými tréninkovými daty. K zařazení nových dat musí být

přečteny předchozí data, což zpomaluje kategorizační proces. Společně s nutností ukládat veškerá data, aby podle nich mohly být kategorizovány nová data, se instance-based learning stává nevhodným pro velké datasety. Výhodou však zůstává jednoduchost tohoto algoritmu.

2.6.4 Model-based learning

Model-based learning je forma učení, kdy ze získaných tréninkových dat vytvoříme model s vhodnými parametry. Nově získaná testovací data vložíme do neuronové sítě. Výchozí hodnoty z neuronové sítě se nachází na vytvořeném modelu, tedy můžeme kategorizovat data, která se již nepodobají tréninkovým datům.

2.6.5 Chybná data

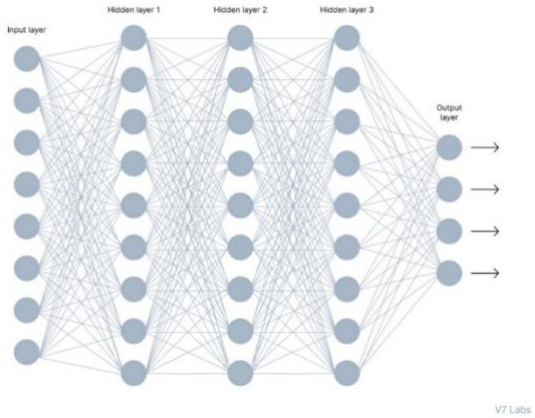
Stejně jako je nutné vytvořit efektivní a výkonný algoritmus schopný zpracování velkého množství dat, je také nutné vkládat do tohoto algoritmu odpovídající data. Příkladem mohou být data se šumem. Tato data sice mají parametry, které může neuronová síť nalézt, ale s větší pravděpodobností dojde k overfittingu tréninkových dat a neuronová síť nebude schopná správně zpracovat testovací data. Dalším typem jsou nepotřebná data. Přestože nenesou žádnou relevantní informaci k nalezení parametrů neuronové sítě, může dojít k nalezení vzorce v těchto datech, který je nevypovídající o chtěném výsledku, nicméně shodou náhod vykazuje korelaci.

2.7 Struktura neuronových sítí

Neuronová síť představuje matematickou počítačovou simulaci vycházející z principu chování mozkových neuronů [2]. Neurony představují početní jednotky provádějící matematické operace, jejichž výstupem jsou zpracované informace, které jsou synapsemi přenášeny do hlubších vrstev sítě. Synapse zároveň zesilují nebo zeslabují přenášenou informaci mezi neurony. Zesílením přenášené informace je udávána důležitost této informace.

Neurony jsou seřazeny do vrstev. První vrstva se nazývá vstupní vrstvou a přijímá vstupní informace. Počet neuronů v první vrstvě je tedy závislý na množině vstupních informací. Následují skryté vrstvy. Počet skrytých vrstev a počet neuronů v nich není pevně daný, nicméně čím více vrstev a neuronů, tím komplexnější problémy jsme schopni za pomoci neuronové sítě řešit. Poslední vrstva se nazývá výstupní a slouží k čtení výstupních dat z neuronové soustavy [2].

Synapse propojují všechny neurony jedné vrstvy se všemi neurony následující vrstvy. Tato metoda propojení se nazývá feed forward. Jde o jednu z nejjednodušších metod propojení neuronů [3].



Obrázek 1 – Struktura neuronové sítě [4]

Ze simulací vidíme, jak každý neuron zpracovává vstupní signály a generuje výstupní signál, který se předává další vrstvě. V důsledku toho se výstupní signál sítě může chovat velmi komplexně a vykazovat vícedimenzionální chování, tzn. že výstup neuronové sítě může být závislý na mnoha faktorech, jako jsou váhy všech neuronů v síti, struktura sítě, velikost vstupů a mnoho dalších proměnných. Tento výstup může být reprezentován v mnohorozměrném prostoru a může vykazovat různé vzorce, které mohou být pro člověka obtížné k pochopení.

V neuronových sítích je základním stavebním kamenem umělý neuron, jehož úkolem je zpracovat hodnoty z neuronů vrchních vrstev. Neurony se skládají z příchozích synapsí, které přináší data ke zpracování. Tyto synapse data zesilují nebo zeslabují podle toho, jakou mají data váhu, proto se synapse označují w (weigth). Neuron sčítá příchozí data vynásobená vahou synapsí a bias b, kterým je zaručena flexibilita výstupu. Dalším důležitým komponentem neuronových sítí jsou aktivační funkce, jež zpracovávají hodnoty, které neuron přenáší výstupními synapsemi hlouběji do neuronové sítě.

Matematický vztah má tento tvar:

$$z(x_i) = f(\sum_{i=1}^{n}(x_i w_i) + b) \tag{2.14}$$

Kde:

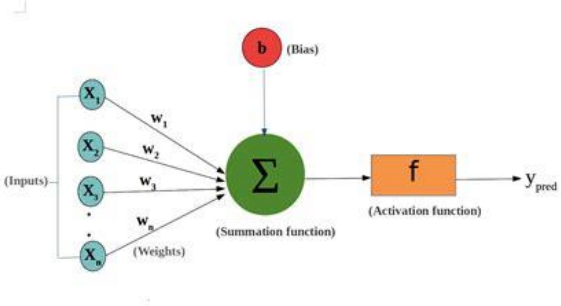
x_i jsou vstupní hodnoty neuronu

w_i jsou váhy synapsí

b je bias

f je aktivační přenosná funkce

z je výstupní hodnota neuronu



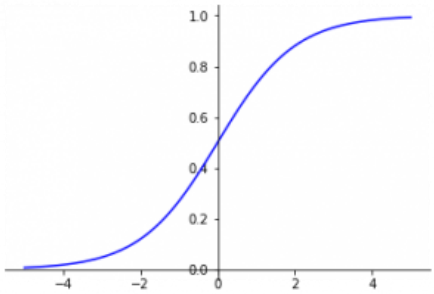
Obrázek 2 –Model neuronu [5]

2.8 Aktivační přenosové funkce

Aktivační funkce jsou klíčovým prvkem neuronových sítí a určují, jakým způsobem budou zpracovávána vstupní data a jak bude síť reagovat na různé stimuly. V této kapitole jsou popsány základní typy funkcí a jejich výhody či nevýhody.

2.8.1 Logistická funkce

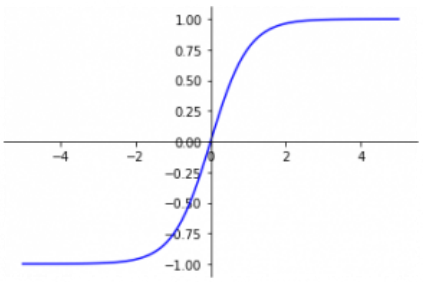
Jelikož se při tvorbě neuronových sítí až do nedávna vycházelo z biologických neuronových sítí, využívalo se výhradně logistické funkce. Tato funkce však omezuje trénink neuronových sítí. Pokud se výstupní hodnoty nevyskytují v blízkosti středu funkce, nacházejí se v oblasti s téměř nulovou derivací.



Obrázek 3 – Logistická (sigmoidní) funkce [6]

2.8.2 Hyperbolický tangent

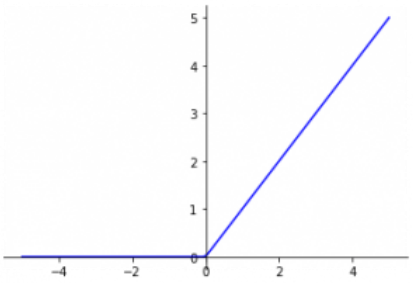
Hyperbolický tangent je funkce podobná logistické funkci, nicméně její střed je v nule, což napomáhá normálovému rozdělení výstupních hodnot z vrstev neuronových sítí, což následně zrychluje trénink funkce. Tato funkce je také omezena nulovým gradientem, nicméně gradient je vyšší ve středu funkce než u logistické funkce.



Obrázek 4 – Funkce tanh [7]

2.8.3 ReLu

Na rozdíl od hyperbolického tangentu má funkce ReLu nenulový gradient pro kladné hodnoty, ale pro záporné hodnoty je gradient roven nule, protože jsou převedeny na nulu. Tento jev může vést k problému tzv. dying ReLU, kdy neurony v neuronových sítích přestanou fungovat a nepodílejí se na přenosu dat. Tento problém lze částečně řešit snížením rychlosti učení [8].



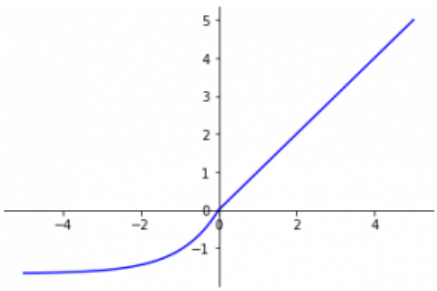
Obrázek 5 – Funkce ReLu [9]

2.8.4 Leaky ReLu

Na rozdíl od standardní aktivační funkce ReLU, Leaky ReLU nezpůsobuje, že záporné hodnoty neuronů padají do oblasti s nulovým gradientem. Tuto funkci lze použít k eliminaci problému dying ReLU a ukazuje se, že použití této funkce zvyšuje výkonnost neuronových sítí. Parametr α, který řídí křivost funkce, lze nastavit na hodnotu α=0,01, což umožní neuronům s negativními hodnotami přejít do stavu s nulovým výstupem a zároveň stále umožňuje jejich probuzení [8]. V případě použití hodnoty α=0,2 se zvyšuje výkon neuronových sítí ještě více [1]. Zvyšováním hodnoty parametru α lze dosáhnout ještě lepší výkonnosti sítě. Nicméně při použití hodnoty α=1 dostaneme lineární funkci, která je sice matematicky jednodušší, ale pro provoz neuronových sítí naprosto nepoužitelná. Pro správný chod neuronových sítí je totiž nutné použít nelineární aktivační funkci.

2.8.5 ELU

Jedná se o nejkomplexnější ale nejúčinnější aktivační funkci. Podobně jako Leaky ReLu má ELU nenulový gradient pro záporné hodnoty, tudíž nedochází k úmrtí neuronů. Další výhodou je hladkost funkce, což urychluje trénink pomocí gradientního sestupu. Výsledkem jsou neuronové sítě se značně sníženým trénovacím časem, avšak kvůli vyšším výpočetním nákladům jsou tyto neuronové sítě při testování datasetu pomalejší než jiné.



Obrázek 6 – Funkce ELU [10]

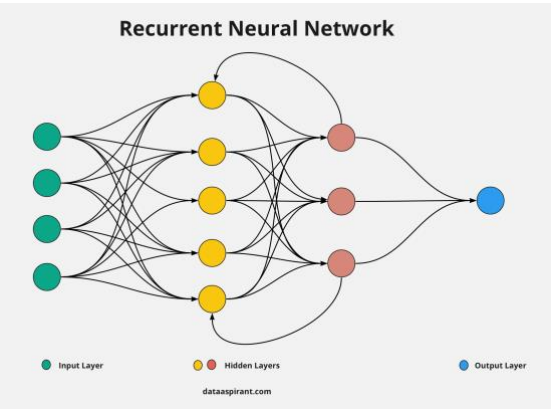
2.9 Rekurentní neuronové sítě

Mnohé dnešní aplikace vyžadují, aby si neuronové sítě byly schopny vzpomenout na již vložená data. Tímto způsobem je například možné vytvořit neuronové sítě schopné rozpoznat řeč, nebo doplnit chybějící slova do vět.

Této paměti v neuronové síti je docíleno vracením informace z neuronů zpět do neuronové sítě, tím je docíleno opakování již vložené informace, což představuje formu paměti. Neuronové sítě, které aplikují tuto strukturu zapojení neuronů se nazývají rekurentní neuronové sítě (RNS).

Jednou z hlavních výhod rekurentních neuronových sítí je možnost zpracování sekvence dat jako je text, video a audio. K plnému porozumění věty, ať psané či řečené, nestačí pouze zpracovat jednotlivá slova, ale je rovněž nutné je vložit do kontextu jako je pořadí slov ve větě atd. Díky rekurentním vazbám rekurentní neuronové sítě ukládají předchozí slova z věty do své krátkodobé paměti a tím získávají kontext celé věty.

Problém u rekurentních neuronových sítí nastává u dlouhých sekvencí dat. Jelikož rekurentní neuronové sítě mají obecně krátkodobou paměť, může dojít ke ztrátě informace ze začátku sekvence. Tento problém je způsobený primárně vanishing gradientem. Dalším problémem je, že přidáním nových rekurentních vazeb do často husté sítě jsou navýšeny výpočetní náklady. Hranice možností se nicméně stále posouvá díky výkonnějším počítačům.



Obrázek 7 – Model rekurentní neuronové sítě [11]

2.9.1 Spectral radius

Jednou z podmínek pro správné fungování reservoir computingu je takzvané stavové zapomínání. Tedy podobně jako u fyzického vodního rezervoáru, nejsou-li vnějšími vlivy vytvářeny vlny, hladina se po nějakém čase ustálí ve stejné výšce jako na začátku. To, jestli k zapomenutí stavu v reservoir computingu dojde, je charakterizováno velikostí spectral radiusu. Speactral radius je největší absolutní hodnota z vlastních čísel matice vah. Pro jistou stabilizaci reservoir computingu musí být velikost spectral radiusu menší než jedna.

2.9.2 Long Short Term Memory (LSTM)

Jak bylo řečeno, krátkodobá paměť u rekurentních neuronových sítí je zásadním problémem, jenž tyto sítě omezuje. Vznikly proto modely, které se tímto problémem zabývají. U struktury LSTM jsou implementovány přepážky tzv. gates, kterými je možno ovlivňovat procházející data. Konkrétně se jedná o forget gate, kterou je určeno, zda informaci zapomenout nebo zachovat, input gate, která rozhoduje, zda aktualizovat důležitost informací, a nakonec je přidána output gate která aktualizuje rekurentní informaci o výstupní hodnotě. Tímto způsobem jsou informace z předchozích kroků zachovány a jsou rovněž dlouhodobě dostupné v krátkodobé paměti, proto long short term memory.

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{2.15}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.16}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.17}$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.18}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{2.19}$$

$$h_t = o_t * tanh(c_t) \tag{2.20}$$

Kde:

x_t je vektor vstupních hodnot v čase t

f_t je vektor forget gatu

i_t je vektor input gatu

o_t je vektor output gatu

h_t je hidden state vektor v čase t

c_t je cell state vektor

$\tilde{c}_t$ je kandidát na nový cell state v čase t

W, U jsou matice vah

2.9.3 Gated Recurrent Unit (GRU)

V modelu GRU je, podobně jako u LSTM, implementována takzvaná reset gate, která v závislosti na nových datech přepisuje hidden state, a update gate, která aktualizuje hidden state o nové informace.

$$\tilde{h}_t = \tanh(W_c[r * h_{t-1}, x_t] + b_c) \tag{2.21}$$

$$u = \sigma(W_u[h_{t-1}, x_t] + b_u) \tag{2.22}$$

$$r = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{2.23}$$

$$h_t = \mathrm{u} * \tilde{h}_t + (1 - u) + h_{t-1} \tag{2.24}$$

Kde:

h_t je hidden state vektor v čase t

$\tilde{h}_t$ je kandidát na nový hidden state v čase t

x_t je vektor vstupních hodnot v čase t

r je vektor reset gatu

u je vektor update gatu

b je bias

W je matice vah

Výhodou GRU oproti LSTM je menší množství parametrů, s čímž souvisí vyšší výpočetní rychlost.

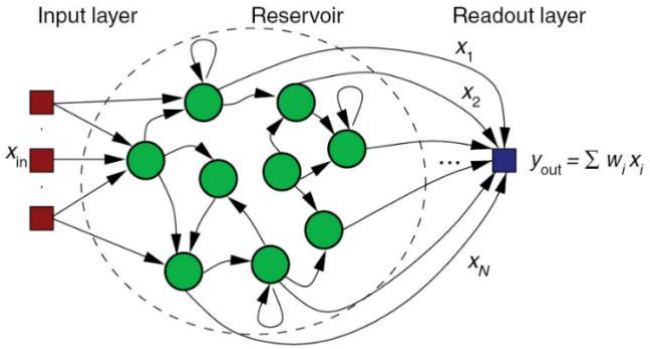
2.10 Reservoir computing

S narůstajícím množstvím neuronů se prodlužuje doba potřebná k tréninku této sítě, což je způsobeno velkým množstvím synapsí mezi neurony.

Cílem reservoir computingu (RC) je snížení počtu dynamických synapsí, což vede k rychlejšímu tréninku neuronových sítí. Využívá se přitom nelineárnosti neuronových sítí vstupu na výstup, způsobenou více dimenzionálním dynamickým systémem, kterým neuronové sítě zpravidla bývají. K největším změnám v hodnotách vah při tréninku neuronových sítí dochází až ve výstupních vrstvách díky algoritmu zpětného šíření. Postup tvorby reservoir computingu je tedy podobný jako u běžných neuronových sítí; vygeneruje se rekurentní neuronová síť s náhodným propojením synapsí a statických vah. Dynamické synapse jsou ponechány jedině ve výstupní vrstvě a na nich je prováděn trénink celé neuronové sítě [12]. Skrytá statická vrstva se stává reservoárem, k němuž se chováme jako k černé skříňce tzv. black box [12]. Velikost rezervoáru je dána počtem neuronů v reservoáru. Obecně platí, že větší počet neuronů může vést k lepším výsledkům, ale také k větší výpočetní složitosti.

Díky rekurentním vazbám, které zachovávají informaci o předchozím stavu, se reservoir computing stává silným nástrojem na předpověď chování chaotických dějů, jako jsou pohyby dvojitého kyvadla, předpověď počasí nebo chování Lorenzova atraktoru [13]. Metoda reservoir computing je však přesná jen po krátkou dobu, pokud nejsou vkládány informace o momentálním stavu. Poté se začnou projevovat chyby, které se akumulují po dobu výpočtu, a model postupně přestává odpovídat realitě. Nicméně výstup systému stále zachovává charakteristické chování chaotického jevu [13].

Reservoir computing se používá v různých oblastech, například v meteorologii pro předpovědi počasí, v robotice pro ovládání robotů, v biomedicíně pro analýzu signálů EKG a EEG a v dalších oblastech. Využití reservoir computingu se neustále rozšiřuje a zdokonaluje, což přináší nové možnosti a využití této metody v různých oblastech vědy a průmyslu.



Obrázek 8 – Model echo state networku [14]

2.10.1 Echo State Network (ESN)

Jedná se o konkrétní typ reservoir computingu, kdy je vygenerována velká síť neuronů s řídkým rekurentním propojením, kde synapse mají pevně dané váhy. Tato síť neuronů tvoří rezervoár, do kterého vkládáme data. Aby byla síť trénovatelná, jsou synapse výstupních neuronů modifikovatelné.

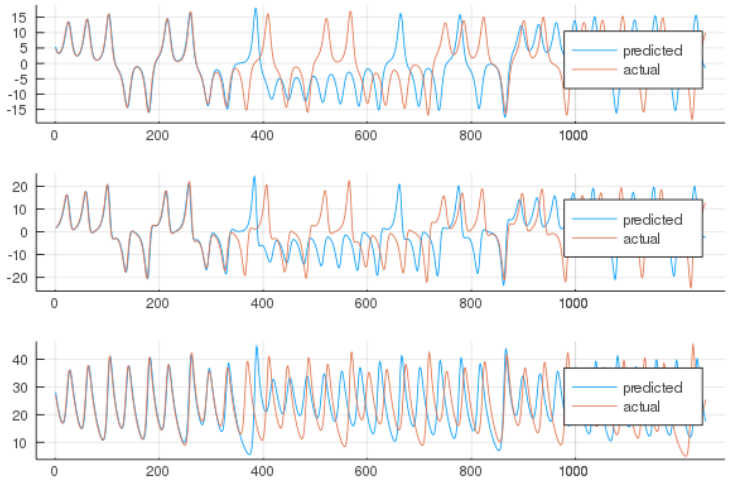
2.10.2 Liquid State Machines (LSM)

Tento typ reservoir computingu se zaměřuje na využití náhodných nebo předem definovaných dynamických systémů (nazývaných "kapaliny"), které se používají jako rezervoár pro zpracování vstupních signálů. Výstupní signál je pak vypočítán pomocí lineární kombinace stavů neuronů v kapalině.

2.11 Využití neuronových sítí

Neuronové sítě mají v dnešní době rozsáhlé pásmo využití. Využívají se v materiálovém průmyslu pro popis chování a vlastností pružných materiálů. V matematice jsou použity pro krátkodobou předpověď chaotických jevů. Využití však můžeme najít i mimo vědecké kruhy. Nejběžněji jsou použity k předpovědi počasí, rozpoznání řeči a předpovědi pohybu na burze. Jak vidíme, spektrum oborů je opravdu rozsáhlé. Důvodem takto rozsáhlého využití je, jak z příkladů můžeme vidět, opakující se problém. Jak můžeme ve zdánlivě chaotickém nelineárním systému určit jeho budoucí stav? Tyto systémy, přestože jsou chaotické, však mají určitý charakter a vzorec chování. To znamená že, pokud budeme mít dostatečné množství příkladů toho, jak se daný systém choval v minulosti, můžeme přibližně určit jeho budoucí stav. Na tomto principu fungují všechny typy neuronových sítí. Velké množství dat je posíláno do neuronových sítí, které vracejí přibližný výsledek. Rozdíl mezi jednotlivými typy neuronových sítí je už pouze v provedení tohoto postupu.

Reálným příkladem využití metody reservoir computing se zabývají v práci *Modeling of Soft Pneumatic Actuators with Different Orientation Angles Using Echo State Networks for Irregular Time Series Data* [15]. Práce se zabývá problematikou předpovědi polohy měkkého pneumatického pohonu. Měkké materiály používané pro pneumatické pohony vykazují při svém pohybu nelinearitu v závislosti na tlaku, kterým jsou naplňovány vzduchové kapsy uvnitř materiálu. Dále měkké materiály vykazují krátkodobou paměť, která vzniká v důsledku poddajnosti měkkého materiálu. Práce se zabývá různými metodami řešení tohoto problému a zaměřuje se na porovnání řešení pomocí Echo State Network (ESN), což je metoda reservoir computingu, a Long Short-Term Memory (LSTM), metoda rekurentní neuronové sítě. Ze získaných výsledků vychází, že LSTM je v tomto problému přesnější než ESN. Nejvýraznější rozdíl přesností obou metod byl pro souřadnici *z*, kde průměrná chyba v předpovědi polohy pro metodu LSTM byla 2,0618 mm, pro metodu ESN byla naměřena průměrná chyba v předpovědi polohy 11.6944 mm. Nejmenší rozdíl naměřených průměrných chyb byl pro souřadnici *y*, chyba předpovědi LSTM byla 3,3739 mm, chyba předpovědi ESN byla 8.2563 mm. Předpověď souřadnic špice měkkého pneumatického pohonu v závislosti na tlaku byla tedy předpovězena pomocí LSTM průměrně (2,5;5,7) -krát přesněji než pomocí ESN. Autoři práce však zmiňují, že přesnost předpovědi ESN by byla vyšší s větším množstvím testovacích dat. Jednou z výhod ESM oproti LSTM je však kratší trénovací doba. Autoři práce dále poukazují, že pro určení polohy *x* špice měkkého pneumatického pohonu, pro který bylo zapotřebí největšího rezervoáru, musel při tréninku LSMT pracovat s 1,085,953 parametry, zatímco ESM pracoval pouze s 2067 parametry. Z měření vyplývá, že je tedy podstatné rozhodnout, zda je pro nás důležitější přesnost předpovědi za cenu delšího tréninku a vyšších výpočetních nákladů, nebo je pro nás podstatnější rychlost a nenáročnost výpočtu za cenu nižší přesnosti získaných dat oproti reálným hodnotám.



Obrázek 9 – Porovnání reálných a vypočtených výsledků [16]

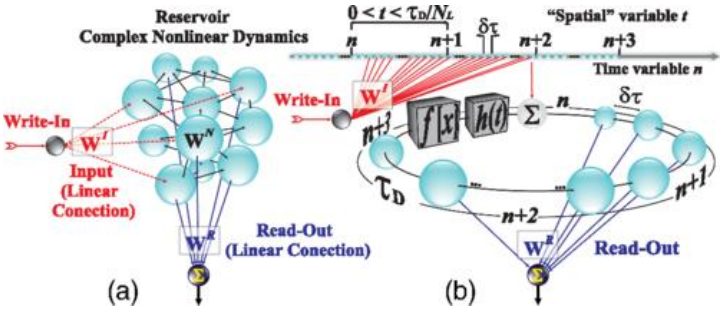
2.12 Metody vycházející z reservoir computingu

Díky svému jednoduchému principu je reservoir computing vhodným systémem pro mnohá vědecká odvětví. Vznikají však nové systémy vycházející z reservoir computingu, které se pokouší o zlepšení charakteristických vlastností reservoir computingu, ať už se jedná o rychlost systému, charakteristické spojení neuronů, nebo zmenšení potřebné velikosti tréninkových dat.

2.12.1 Delayed Reservoir Computing (delay-RC)

Jednou z možností, jak zlepšit výpočetní rychlost neuronových sítí, je převedení počítačového modelu do reálného elektrického nebo fotoelektrického obvodu, což by ovšem vyžadovalo převedení všech neuronů dané neuronové sítě do jednotlivých obvodů, jež by bylo nutné spolu náhodně propojit. Jedná se však o příliš zdlouhavý a nákladný proces. Statická dynamika rezervoáru nabízí alternativní způsob řešení. Jelikož je rezervoár neměnný, jsme schopni nahradit celý rezervoár jedním neuronem s jednou rekurentní smyčkou. Jedná se tedy o dynamický nelineární systém, který funguje s časovým zpožděním, proto se tato metoda nazývá delayed reservoir computing (delay-RC). Tyto systémy po správném zvolení parametrů vykazují stejnou dynamičnost jako standartní reservoir computing [17]. Trénink delay-RC probíhá stejně jako u standartního reservoir computingu. Zde jsou však neurony, ze kterých výstupní vrstva získává data, virtuální a nedochází na nich k žádné přeměně dat. Přeměna dat probíhá právě na jednom reálném neuronu. Místo toho představují virtuální neurony hodnoty získané z reálného neuronu v různých časových intervalech [18].

V praxi se pro realizaci zpoždění využívá dostatečně dlouhého optického kabelu. Tato metoda našla využití v elektronice, fotoelektronice a fotonice a řeší problémy jako rozpoznání hlasu a obličeje.



Obrázek 10 – RC (vlevo) a delay-RC (vpravo) [19]

2.12.2 New Generation Reservoir Computing (NG-RC)

Přestože je reservoir computing rychlým a datově nenáročným systémem, jednou z jeho nevýhod je právě náhodné propojení neuronů v rezervoáru. To způsobuje různě kvalitní výsledky pro stejná data, získaná z různých systémů reservoir computingu. Přestože můžeme říci, do jaké míry je daný reservoir computing systém optimalizován, nejsme schopni určit přesné spojení neuronů za účelem replikace této optimalizace na jiných systémech reservoir computingu**.** Autoři práce *Next generation reservoir computing* [20] však poukazují na to, že za určitých podmínek je dobře optimalizovaný reservoir computing systém ekvivalentní systému nelineární vektorové autoregrese (NVAR). Velkou výhodou NVAR systému je absence rezervoáru, čímž se značně omezuje náhodnost výsledků a zvyšuje se výpočetní rychlost systému. Autoři práce [20] tento systém nazvali novou generací rezervoár computingu (NG-RC). NG-RC je díky své optimalizaci schopen dávat stejné výsledky jako tradiční reservoir computing, ale vyžaduje k tomu menší množství vstupních dat. Zároveň má NG-RC méně výstupních synapsí a je tedy zapotřebí menšího času k tréninku systému NG-RC oproti tradičnímu reservoir computing systému.

2.12.3 Quantum Reservoir Computing (QRC)

Další oblastí, kde se využívá rychlého tréninku reservoir systému je oblast kvantových počítačů (quantum computing). Kvantové počítače jsou založeny na práci s kvantovým bitem (qubit). Protože se qubit jeví jako zároveň 0 a 1 pomocí principu superpozice [21], značně se tím navyšuje výpočetní rychlost, a to exponenciálně s počtem qubitů. Nicméně zásadním problémem jsou extrémní podmínky, za kterých mohou qubity existovat [22]. Dnešní kvantové počítače ještě nejsou natolik výkonné, aby nahradily běžné bitové počítače. Nicméně počet qubitů v kvantových počítačích se stále navyšuje.

Quantum Reservoir Computing (QRC) využívá vlastností qubitů, které navyšují volné parametry systému exponenciálně s počtem qubitů. Tím získáváme flexibilní systém s výhodou rychlého tréninku. Ve své práci [23] P. Mujal využívá QRC pro předpověď energetické hladiny kvantové částice v závislosti na potenciálu skvrny (speckle potential). Byly použity dva QRC systémy pro porovnání výsledků. Jeden systém pracoval pouze s jedním qubitem a 61 volnými parametry. Druhý systém počítal s dvěma qubity a 451 volnými parametry. Bylo použito 10000 časově diskrétních dat. 7500 dat bylo použito na trénink obou systémů QRC, na zbylých 2500 byly systémy testovány. Úspěšnost modelů byla měřena průměrnou absolutní chybou (MAE) a koeficientem determinace (R^2). Koeficient determinace nám udává, jak moc jsou na sebe předpovězené a reálné hodnoty lineárně závislé ($R^2 \epsilon < 0{,}1 >$). Ukázalo se, že i systém s jedním

qubitem a 61 volnými parametry se dokázal rychle naučit z trénovacích dat bez toho, aby došlo k overfittingu. Parametry prvního systému byly MAE = 0,1126 a R^2= 0,8808. Výsledky druhého systému byly MAE = 0,0802 a R^2= 0,916.

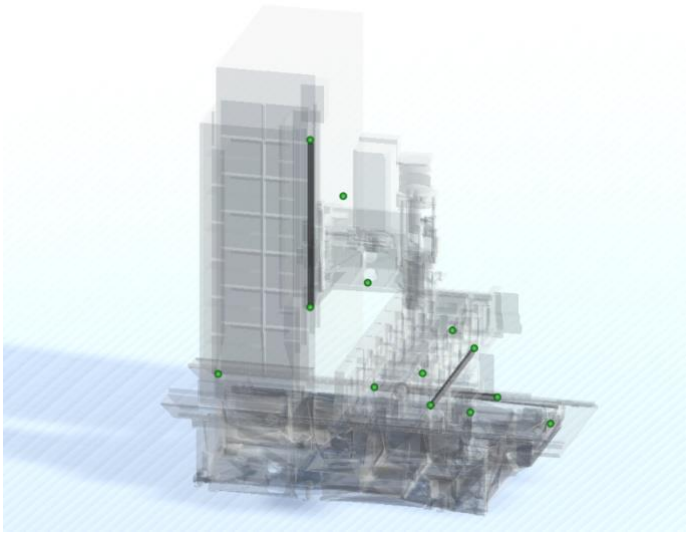
Z měření vyplývá, že QRC je vhodným systémem pro předpověď energie kvantové částice pomocí potenciálů skvrnité poruchy, a to s dobrou přesností. Dále P. Mujal polemizuje nad dalším zefektivněním QRC. Se zvětšením počtu spinů kvantové částice nebo s větším počtem kvantových částic by volné parametry QRC vzrostly exponenciálně, to by vedlo k více flexibilnímu systému s větší pamětí i v relativně malém systému.

3 Implementace reservoir computingu na reálných datech

V praktické části této bakalářské práce se zaměřujeme na realizaci návrhu a implementaci řešení pro zpracování a analýzu dat. Konkrétně řešíme, jaký typ dat je vhodné zpracovat a jaký hardware a software je vhodné použít. Vzhledem k charakteru kooperativní práce a možnému budoucímu nasazení této metody do průmyslové výroby je kód utajen. Součástí této kooperativní práce je však trénovací algoritmus, kterým je popsán postup trénování neuronových sítí. Kód k tomuto algoritmu je dostupný na GIT ústavu UVSSR.

3.1 Odběr dat

Data, která slouží k trénování neuronové sítě, jsou odebírána z výrobní buňky, konkrétně se jedná o logická data typu true/false. Dále je sledováno zda je stroj v provozu, zda jsou zajištěné dveře, zda nedošlo k chybě v programu atd. Rovněž jsou odebírána data o teplotě a poloze operujících částí stroje, jako například motory pro osy XYZ, ložiska na daných osách, teplota elektrického rozvaděče, a teplota v okolí stroje (viz. Obrázek 11).



Obrázek 11 – Výrobní buňka s vyznačenými body měření teploty

3.2 Redukce dat

Přestože je reservoir computing rychlá a efektivní metoda pro zpracování velkého množství dat, jsou pro nás jistá získaná data zbytečná. Primárně se jedná o data, ve kterých nedochází k žádné změně. Reservoir se může učit pouze ze změny hodnoty. Jelikož jsou data z reálného průmyslového zařízení, jsou přítomny senzory, ze kterých nejsou využity žádná data. Redukce dat musí proběhnout takovým způsobem, aby výsledek získaný z neuronové sítě byl stejný, nebo velmi podobný, jako kdybychom používali nezredukovaná data [24].

3.2.1 Způsoby redukce dat

Dimenzionální redukce dat je soubor metod založených na vyhodnocení relevantních dat a zanedbání zbytku datasetu. Tímto způsobem redukujeme počet proměnných, se kterými

pracujeme. Menší dataset je pro nás důležitý, jelikož odstraněním přebytečných dat zrychlíme výpočetní operace [25].

- Vlnová transformace je metoda, která nahrazuje vektor dat X na vektor stejné délky X´. Tento nový vektor už můžeme zkrátit o nepotřebná data [26].

- Analýza hlavních komponentů je metoda založená na vyhodnocení parametrů s největším rozptylem. Tyto data jsou pro nás nejvíce vypovídající, proto je žádoucí je zachovat, zatímco zbylá data v malém rozptylu nenesou pro nás nikterak podstatnou informaci [27].

- Metoda výběru podmnožiny atributů prochází dataset a vyhodnocuje jeho parametry. Parametry, které nepřináší žádnou použitelnou nebo významnou hodnotu, odstraňuje, a naopak parametry nejvíce vypovídající o datasetu ponechává [28]. Tento proces se opakuje, dokud nedostaneme dataset s proměnnými o námi zvolené důležitosti [29].

Cílem redukce početnosti dat je nahrazení parametrů pouze jejich atributy.

- Parametrická redukce dat předpokládá, že dataset se drží jistého modelu. Tento model obsahuje nezávislé parametry, které daný model definují. Tato metoda se sestavuje data z datasetu tak, abychom zjistili parametry tohoto modelu, což znamená že data jsou lineárně závislá na proměnné. Redukci dat poté provedeme lineární regresí.

- Neparametrická redukce dat nepředpokládá návaznost dat. Zredukovaná data jsou větší než u parametrické redukce. Jednotlivé metody zahrnují:

    - Histogram, který nám udává, s jakou frekvencí se veličiny v dané proměnné vyskytují.

    - Shlukování, s jehož pomocí jsou datům přiřazeny atributy v závislosti na tom, jak blízko jsou k datům se stejnými atributy.

    - Vzorkování, kdy namísto práce s celým data setem, vybereme z určitého parametru pár náhodně vybraných vzorků, jež tento dataset budou reprezentovat.

    - Agregace datové kostky, při níž redukce spočívá v zjednodušení informace z více datasetů a tato zjednodušená informace je vložena do nového datasetu.

    - Komprese dat, jež zajišťuje optimalizaci datové velikosti, kterou dataset zabírá. V této formě však není možné s ním pracovat a zredukovaná data musí být obnovena. V závislosti na ztrátě dat se tento proces nazývá bezeztrátová komprese a ztrátová komprese. Ztrátovou kompresí mohou trpět primárně obrázky, videa a zvukové soubory. S každou ztrátovou kompresí se snižuje kvalita souboru.

3.3 Hardware

Hardware hraje velmi důležitou roli v oblasti reservoir computingu. Pro reservoir computing jsou ideální hardwarové platformy, které umožňují vysoký výkon a paralelizaci výpočtů na více jádrech. Kombinace takového hardwaru s dobře navrženým softwarem může vést k velmi efektivnímu a rychlému trénování neuronových sítí.

3.3.1 NVIDIA Jetson AGX XAVIER

NVIDIA Jetson AGX Xavier je výkonný výpočetní modul určený pro použití v aplikacích umělé inteligence a strojového učení. Jedná se o jednu z nejvýkonnějších platforem pro zpracování dat v reálném čase a podporuje širokou škálu vstupních zařízení, jako jsou kamery, senzory a mikrofony [30].

Tento modul je vybaven osmi jádry procesoru ARM a 512 jádry NVIDIA Volta Tensor Cores, což mu umožňuje dosáhnout výkonu až 32 TOPS (trilión operací za sekundu). Díky tomu je schopen zpracovávat velké objemy dat a provádět složité výpočetní úlohy, jako je rozpoznávání obrazů, zpracování řeči a analýza dat [30].

Jetson AGX Xavier také podporuje různé rozhraní pro připojení periferních zařízení, jako jsou Ethernet, HDMI, USB a PCIe. To umožňuje připojení dalších zařízení a rozšíření funkcionality platformy [30].

Jetson AGX XAVIER poskytuje výkon a paměťovou kapacitu, které jsou nutné pro trénování a používání rezervoárů v reálném čase. Díky podpoře CUDA a Tensor Cores může Jetson AGX XAVIER urychlit výpočetní operace, což zlepšuje výkon trénování [30].

Navržené softwarové řešení, které je provozováno na tomto hardwaru, bude popsáno v kapitole Software.

3.3.2 Datový server a databáze MongoDB

MongoDB je NoSQL databázový systém, který lze využít v reservoir computingu pro ukládání, správu a načítání velkého objemu dat. Výhodou MongoDB je schopnost pracovat s nestrukturovanými daty, což je často případ v oblasti strojového učení a neuronových sítí. MongoDB je navržen tak, aby umožňoval snadnou horizontální škálovatelnost a replikaci dat na více serverů pro zajištění vysoké dostupnosti a spolehlivosti [31].

Měřená data jsou ukládána do databáze a tříděna podle denních záznamů. Jeden denní záznam představuje přibližně 2Gb data, které představují data z měření různých veličin v celém technologickém celku. Z důvodu nemožnosti veřejně publikovat detaily měření je prezentace dat v této práci omezena jen na reprezentativní příklady.

3.3.3 Fast data storage

Fast data storage je termín používaný pro popis rychlého ukládání dat, které jsou potřebné pro rychlé zpracování a analýzu v reálném čase. V případě tvorby neuronových sítí je rychlý přístup k datům velmi důležitý, protože čím rychleji jsou data načtena, tím rychleji může být síť trénována a zpracovávána v reálném čase.

Pro ukládání dat z MongoDB databáze jsou použity čtyři desky Arduino s 2TB SSD disky jako fast data storage, které jsou následně použity jako vstup do reservoiru na NVIDIA

Jetson AGX XAVIER. Tyto disky mohou být využity pro rychlé ukládání dat a zajištění rychlého přístupu k nim.

V průběhu řešení této práce byla jedna jednotka zničena (pravděpodobně v důsledku přehřátí), nicméně zbývající tři zůstávají funkční. Data na zničeném nosiči jsou duplikována na jiných serverech, takže nedošlo ke ztrátě informací, nicméně způsobilo zdržení v realizační práci, protože uchovávala mezivýsledky výpočtů.

3.4 Software

Reservoir computing vyžaduje efektivní zpracování a přesun velkého množství dat. Je proto důležité mít k dispozici vhodný software, který umožní správu dat a řízení toku dat mezi jednotlivými komponentami systému. Je tedy klíčové vybrat a správně nakonfigurovat software, který bude umožňovat rychlou a spolehlivou manipulaci s daty v celém procesu.

3.4.1 Navržené softwarové řešení – základní přehled

Navržené softwarové řešení sestává z několika dílčích softwarových řešení

- **Implementace Reservoir computingu**
    - HW: NVIDIA Jetson
    - SW Komunikace: Nastavení a řízení trénování sítě přes REST API; Publikování výsledků výpočtů přes Apache Kafka (jako Publisher).
- **Fast Data Storage SW:**
    - HW: Arduino s SSD disky
    - SW komunikace: Consumer ze systému Apache Kafka; REST API pro načítání dat pro jiné zdroje.
- Pro **řízení toku dat** je použit Apache Nifi
- Pro ukládání data **databáze MongoDB**
- Pro **datový stream** byl použit SW Apache Kafka

3.4.2 Navržené softwarové řešení – použité technologie

Pro softwarové řešení se ukázalo jako výhodné (zejména z časových důvodů) použít dostupné softwarové řešení, zejména v té částí softwarového řešení, jako je ukládaní dat nebo řízení toku dat.

3.4.2.1 Řízení toku dat – Apache Nifi

Apache NiFi je open source nástroj pro vizuální návrh a správu datových toků. Je vhodný pro integraci dat z různých zdrojů, transformaci dat a přenos dat mezi různými systémy. V reservoir computingu lze využít Apache NiFi pro sběr dat z různých senzorů, předzpracování dat a jejich transformaci pro vstup do rezervoárového výpočtu [32].

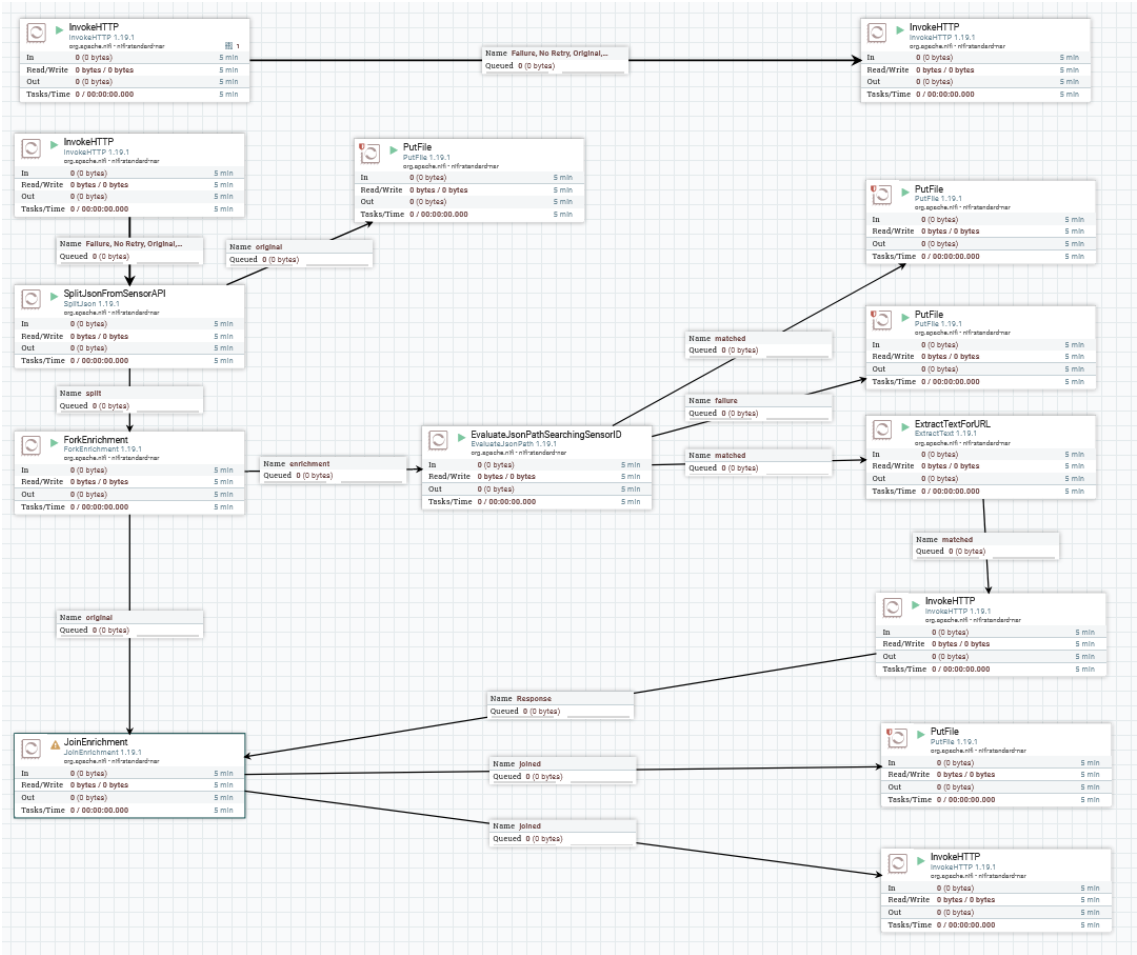
Konkrétně lze v Apache NiFi vytvořit dataflow pro sběr dat z různých senzorů, například teplotních senzorů, a pro jejich následné předzpracování a transformaci pro vstup do rezervoárového výpočtu. Apache NiFi také umožňuje integraci s různými

databázovými systémy, což je užitečné pro ukládání a načítání dat potřebných pro rezervoárový výpočet [32].

Dalším využitím Apache NiFi v reservoir computingu je možnost integrace s Apache Kafka, což umožňuje streamování dat přímo do rezervoárového výpočtu. To může vést k výraznému zrychlení výpočtu a zlepšení výkonu [32].

Apache NiFi nabízí grafické rozhraní, které umožňuje uživatelům vytvářet, konfigurovat a spravovat datové toky. Tyto toky mohou být vytvořeny pomocí předdefinovaných procesorů, které umožňují manipulaci s daty (např. filtrování, transformace, agregace, validace atd.) a také pomocí vlastních procesorů, které lze implementovat pomocí jazyků Java, Groovy, Python nebo Ruby [32].

NiFi má také rozsáhlou knihovnu nástrojů pro integraci s různými systémy, včetně Apache Kafka, MongoDB, a dalších. Díky tomu lze Apache NiFi snadno integrovat s existujícími infrastrukturami a vytvářet robustní datové toky, které zajišťují spolehlivost, škálovatelnost a výkonnost [32].



Obrázek 12 – Příklad implementace Apache Nifi

3.4.2.2 Datový stream – Apache Kafka

Apache Kafka je distribuovaný streamovací a publish-subscribe systém, který umožňuje rychlý a spolehlivý tok dat mezi různými systémy a aplikacemi. Díky své struktuře

a schopnosti zpracovávat velké množství dat v reálném čase je Apache Kafka vhodný pro využití v reservoir computingu.

Konkrétně lze Apache Kafka použít jako prostředek pro sběr a distribuci dat mezi různými vstupními zdroji a výstupními cíli reservoiru. Vstupní data mohou být sbírána z různých zdrojů pomocí Kafka Connect a distribuována do reservoiru pomocí Kafka Streams. Výstupní data lze poté distribuovat do různých cílových systémů pomocí Kafka Connect.

V navržené aplikaci Apache Kafka slouží ke streamování data mezi jednotlivými podsystémy.

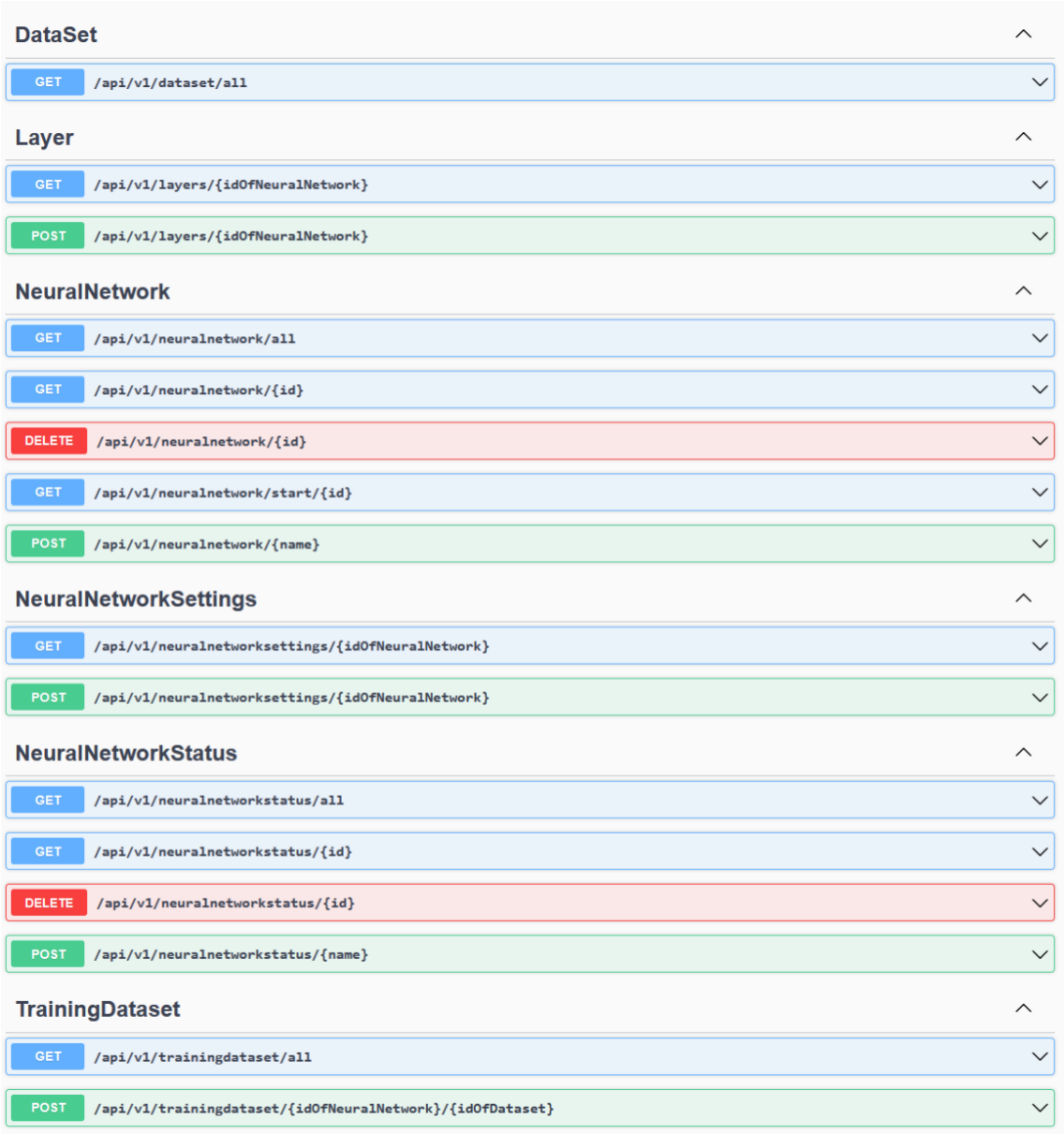
3.4.2.3 REST API

REST (Representational State Transfer) API je architektura pro vytváření webových služeb, která umožňuje komunikaci mezi různými aplikacemi prostřednictvím standardních HTTP požadavků. REST je navržen tak, aby byl jednoduchý, robustní a snadno škálovatelný.

REST API pracuje s kolekcemi zdrojů, jako jsou například záznamy v databázi, které jsou identifikovány pomocí unikátního URI (Uniform Resource Identifier). Každý zdroj může být modifikován pomocí standardních HTTP metod jako GET, POST, PUT a DELETE, které umožňují získávat, vytvářet, aktualizovat a mazat data.

REST API vrací data v různých formátech, jako jsou například JSON, XML nebo HTML, což umožňuje jednoduchou integraci s jinými aplikacemi.

V rámci práce bylo využito interaktivního prostředí Swagger UI, které umožňuje procházet a testovat koncové body API softwarových aplikací. Pomocí tohoto interaktivního prostředí jsou sítě přednastavovány a trénovány. Dále jsou sbírány informace o jejich stavu.



Obrázek 13 – Navržené REST API provozované na HW Nvidia jetson:

3.5 Trénovací algoritmus

Jako první krok je nutné připravit data získaná z výrobní buňky a přiřadit k nim již zjištěná cílová data o přesnosti stroje.

Pro zpracování neuronovou sítí je třeba převést vstupní data na jednotný číselný formát, jako například informaci o stavu true/false, on/off převést na 1/0. V tomto kroku může být využit například Apache Kafka pro přenos dat v reálném čase. Tato upravená data jsou uložena na server.

Jsou přednastaveny hyperparametry sítě, jako je velikost reservoiru, výběr řešiče pro výpočet vah reservoiru, výběr regularizačního parametru a škálování vstupních dat.

Pro každou dávku dat je provedeno dopředné šíření sítě, jehož výstupem jsou predikované hodnoty. Tyto výsledky slouží k výpočtu ztrátové funkce a nalezení minima ztrátové funkce pomocí zvoleného řešiče.

Výsledkem gradientního sestupu je aktualizace výstupních vah. Tento proces je opakován, dokud nedojde k dosažení oblasti minima ztrátové funkce.

Po dokončení trénování je třeba testovat výkon rezervoáru na testovacích datech, aby byl ověřen jeho výkon. Pokud je výkon reservoiru nedostačující, je třeba provést trénink znovu, nebo znovu za použití jiných hyperparametrů.

Po úspěšném trénování by síť měla být schopna s jistou tolerancí předpovídat výstupy pro nová vstupní data. Výstupem trénovacího algoritmu je tedy sada vah, která umožňuje neuronové síti vypočítávat výstupy pro dané vstupy. Tyto váhy mohou být následně použity pro testování sítě na nových vstupech a pro porovnání předpovědí s očekávanými výstupy.

Kód k tomuto algoritmu je dostupný na GIT ústavu UVSSR

3.5.1 Testovací matice

Popis experimentu je vyjádřen testovací maticí parametrů k ověření funkčnosti navrženého řešení.

- Aktivační funkce: logistická, sigmoidní, Tanh, Identity, rectliní, HardSigmoid, LeakyReLU
- Počet neuronů: 1-3 skryté vrstvy s počtem neuronů od 200 po 20 do 300, pro každé nastavení se mění velikost předposlední vrstvy
- Délka tréninku: trénování limitováno na 150 epoch
- Data: pro trénink byla použita data o teplotě a stavu stroje zapnuto/vypnuto za celý poslední rok

Celý trénink testovací matice zabral 17 dní.

3.5.2 Výsledná data – tabulky

Kvůli velkému množství dat, jsou na ukázku přidány alespoň tři tabulky vypočtených dat. Tyto tabulky s vypočtenými hodnotami $RMSE$ se liší použitými aktivačními funkcemi, jež jsou popsány výše, a rovněž se liší ve velikosti třetí skryté a předposlední vrstvy,

na níž je prováděn trénink daného reservoiru. Podle hodnot $RMSE$ budou následně zhodnocena jednotlivá nastavení sítí.

Tabulka 1: Část výsledných hodnot pro aktivační funkci ReLu

AF	Relu			
Layer1	Layer2	Layer3	LastLayer	RMSE
200	200	200	20	0,95764
200	200	200	40	0,855706
200	200	200	60	0,737859
200	200	200	80	0,609449
200	200	200	100	0,539621
200	200	200	120	0,466558
200	200	200	140	0,365893
200	200	200	160	0,283044
200	200	200	180	0,175675
200	200	200	200	0,093361

Tabulka 2: Část výsledných hodnot pro rozšířenou síť s aktivační funkcí ReLu

AF	Relu			
Layer1	Layer2	Layer3	LastLayer	RMSE
200	200	260	20	0,176144002
200	200	260	40	0,17726994
200	200	260	60	0,179680222
200	200	260	80	0,182587117
200	200	260	100	0,18000242
200	200	260	120	0,17754659
200	200	260	140	0,183595067
200	200	260	160	0,182975839
200	200	260	180	0,17433354
200	200	260	200	0,176739373

Tabulka 3: Část výsledných hodnot pro rozšířenou síť s aktivační funkcí Leaky ReLu

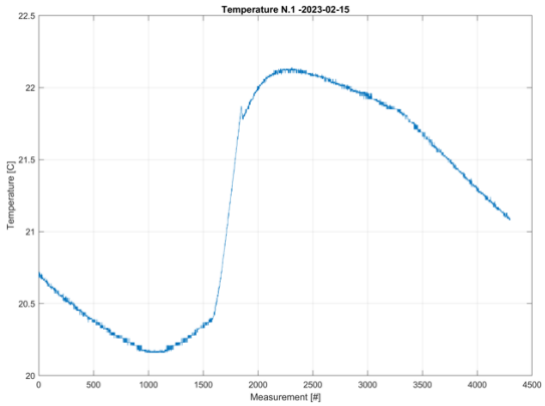
AF	LeakyRelu			
Layer1	Layer2	Layer3	LastLayer	RMSE
200	200	260	20	0,939957434
200	200	260	40	0,826460787
200	200	260	60	0,727492594
200	200	260	80	0,64442879
200	200	260	100	0,554168082
200	200	260	120	0,47702701
200	200	260	140	0,386351644
200	200	260	160	0,278127202
200	200	260	180	0,18862397
200	200	260	200	0,094997198

Z tabulek vidíme korelaci mezi počtem neuronů v poslední vrstvě a velikostí hodnoty $RMSE$. Větší množství neuronů v poslední vrstvě přináší přesnější výsledky. Je nutné poukázat na méně přesnou síť v tabulce 2. Přestože má tato síť více neuronů, je méně přesná; toto může být způsobeno náhodným propojením reservoiru. Je možné, že vygenerování sítě se stejnými parametry by natrénovalo síť přesnější než v tabulce 1.

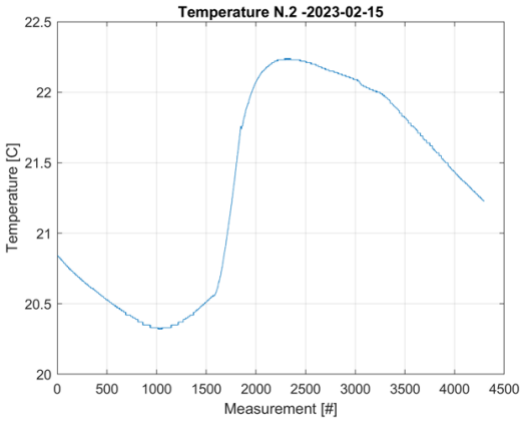
3.5.3 Výsledná data – grafy

Podobně jako u tabulek i grafů je velké množství vzhledem k velkému počtu senzorů měřící teplotu. Grafy zobrazují vzhled měřených dat pro dny, kdy stroj nebyl. nebo byl zapnutý a následující grafy ukazují predikci dat ze zapnutého stroje.

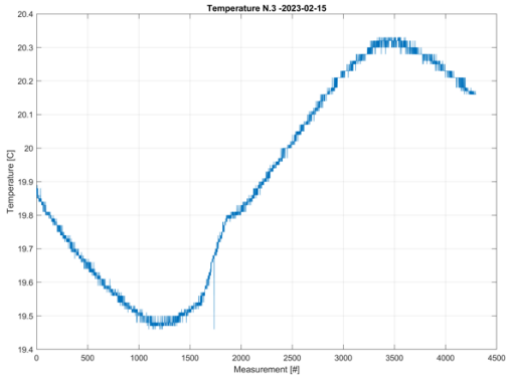
Následující grafy měření zachycují typický vzhled dat z pěti senzorů výrobní buňky pro den, kdy nebyl stroj zapnutý.



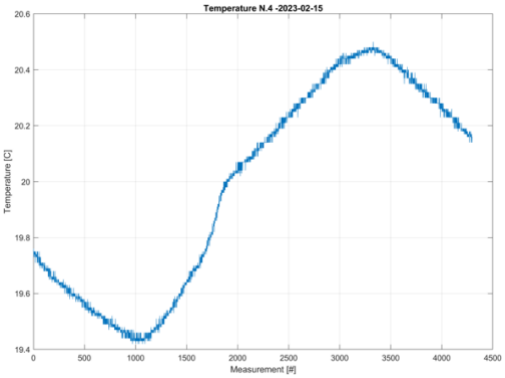
Obrázek 14: Naměřené hodnoty ze senzoru 1 pro 15.2.2023



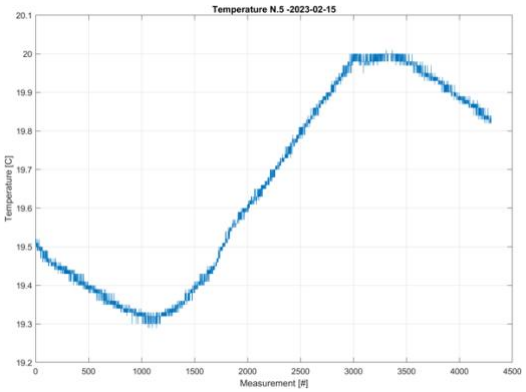
Obrázek 25: Naměřené hodnoty ze senzoru 2 pro 15.2.2023



Obrázek 36: Naměřené hodnoty ze senzoru 3 pro 15.2.2023

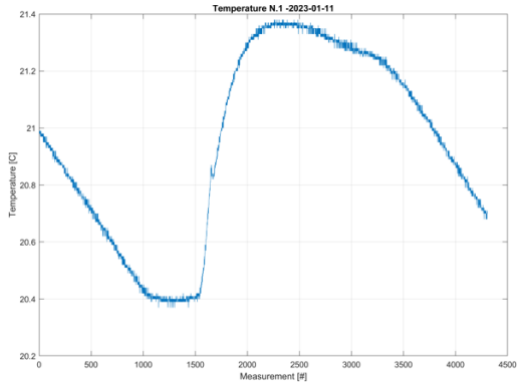


Obrázek 47: Naměřené hodnoty ze senzoru č pro 15.2.2023

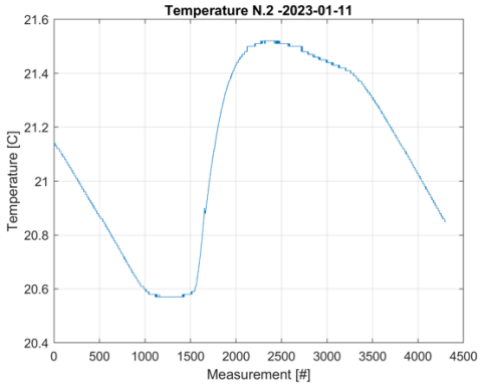


Obrázek 58: Naměřené hodnoty ze senzoru 5 pro 15.2.2023

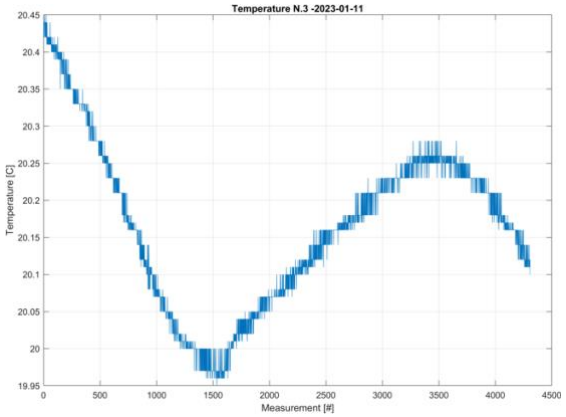
Následující grafy obsahují hodnoty měřené teploty ze dne, kdy byl stroj zapnutý a byly provedeny predikce pro tyto senzory.



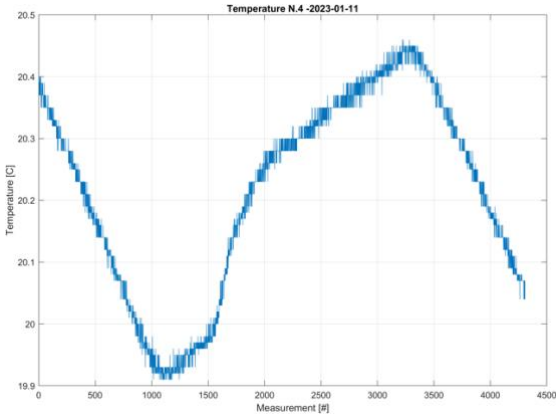
Obrázek 69: Naměřené hodnoty ze senzoru 1 zapnutého stroje pro 11.1.2023



Obrázek 20: Naměřené hodnoty ze senzoru 2 zapnutého stroje pro 11.1.2023

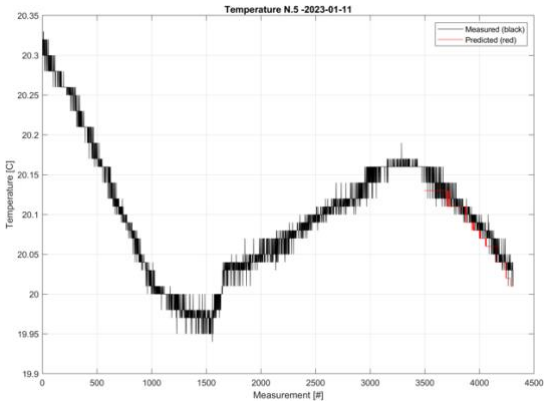


Obrázek 21: Naměřené hodnoty ze senzoru 3 zapnutého stroje pro 11.1.2023

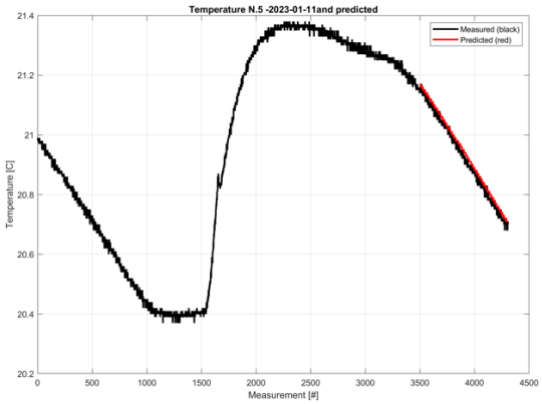


Obrázek 22: Naměřené hodnoty ze senzoru 4 zapnutého stroje pro 11.1.2023

Poslední dva grafy obsahují vykreslené hodnoty senzorů a predikce těchto hodnot ze dne kdy byl stroj zapnutý.



Obrázek 23: Naměřené a předpovězené hodnoty ze senzoru 5 zapnutého stroje pro 11.1.2023



Obrázek 24: Naměřené a předpovězené hodnoty ze senzoru 1 zapnutého stroje pro 11.1.2023

Z grafů vidíme shodu predikovaných dat, metodou reservoir computing jsme tedy schopni s vysokou přesností predikovat teploty stroje v provozu.

4 Závěr

V práci byla provedena rešerše z oblasti reservoir computingu a jeho využití v různých typech neuronových sítí. V rámci rešerše byla prozkoumána problematika střední kvadratické chyby, gradientního sestupu, regularizovaných lineárních modelů a typů systémů strojového učení, včetně struktury a aktivačních přenosových funkcí neuronových sítí a rekurentních neuronových sítí. Byly rozebrány metody trénování neuronových sítí, aktivační přenosové funkce a struktura neuronových sítí. Dále byly podrobněji vysvětleny rekurentní neuronové sítě a její specifika, jako LSTM nebo GRU a spectral radius.

Hlavním tématem práce byl pak reservoir computing. Byly rozebrány základní metody Echo State Network a Liquid State Machines, ale také novější metody jako Delayed Reservoir Computing, New Generation Reservoir Computing a Quantum Reservoir Computing.

V praktické části práce byl navržen a implementován trénovací software s využitím Apache Nifi, Apache Kafka, REST API a databáze MongoDB. Testování softwarového řešení bylo provedeno na reálných datech získaných z měření sledované technické soustavy. Výsledky trénování byly následně porovnány a zhodnoceny. Hlavním výsledkem měření je potvrzení možnosti použití reservoir computingu pro průmyslové aplikace, a to díky vysoké přesnosti, se kterou jsou průmyslová data predikována.

Tato bakalářská práce může sloužit jako ucelený zdroj informací pro ty, kteří se zajímají o tuto problematiku a chtějí se dozvědět více o možnostech využití neuronových sítí a reservoir computingu v praxi.

5 Seznam použitých zdrojů

[1] GÉRON, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems [online]. nedatováno [vid. 2023-05-21]. Dostupné z: https://books.google.com/books/about/Hands_On_Machine_Learning_with_Scikit_Le.html?hl=cs&id=HHetDwAAQBAJ

[2] *Activation Functions in Neural Networks [12 Types & Use Cases]* [online]. [vid. 2023-05-21]. Dostupné z: https://www.v7labs.com/blog/neural-networks-activation-functions

[3] *Feed Forward Neural Network Definition | DeepAI* [online]. [vid. 2023-05-21]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[4] *60d242974bcba9f8c670e03e_Group 806.jpg (1094×832)* [online]. [vid. 2023-05-21]. Dostupné z: https://assets-global.website-files.com/5d7b77b063a9066d83e1209c/60d242974bcba9f8c670e03e_Group%20806.jpg

[5] *1*Bo2gNRKR3wDHzFsseDvMeg.png (700×385)* [online]. [vid. 2023-05-21]. Dostupné z: https://miro.medium.com/v2/resize:fit:700/1*Bo2gNRKR3wDHzFsseDvMeg.png

[6] *Sigmoid-300x206.png (300×206)* [online]. [vid. 2023-05-21]. Dostupné z: https://inside-machinelearning.com/wp-content/uploads/2021/02/Sigmoid-300x206.png

[7] *Tanh-300x199.png (300×199)* [online]. [vid. 2023-05-21]. Dostupné z: https://inside-machinelearning.com/wp-content/uploads/2021/02/Tanh-300x199.png

[8] REK, Petr. Knihovna pro návrh konvolučních neuronových sítí [online]. nedatováno [vid. 2023-05-22]. Dostupné z: http://dspace.vutbr.cz/handle/11012/84982

[9] *ReLU-300x205.png (300×205)* [online]. [vid. 2023-05-21]. Dostupné z: https://inside-machinelearning.com/wp-content/uploads/2021/02/ReLU-300x205.png

[10] *ELU_-300x199.png (300×199)* [online]. [vid. 2023-05-21]. Dostupné z: https://inside-machinelearning.com/wp-content/uploads/2021/02/ELU_-300x199.png

[11] *3-Recurrent-Neural-Network.png (1704×1262)* [online]. [vid. 2023-05-21]. Dostupné z: https://i1.wp.com/dataaspirant.com/wp-content/uploads/2020/11/3-Recurrent-Neural-Network.png?ssl=1

[12] NAKAJIMA, Kohei a Ingo FISCHER. *Natural Computing Series Reservoir Computing Theory, Physical Implementations, and Applications* [online]. nedatováno. Dostupné z: http://www.springer.com/series/4190

[13] *A brief introduction to Reservoir Computing | Francesco Martinuzzi* [online]. [vid. 2023-05-21]. Dostupné z: https://martinuzzifrancesco.github.io/posts/a-brief-introduction-to-reservoir-computing/

[14] *reservoir_components.png (890×488)* [online]. [vid. 2023-05-21]. Dostupné z: http://jujuba.me/imgs/reservoir_components.png

[15] YOUSSEF, Samuel M., Mennaallah SOLIMAN, Mahmood A. SALEH, Mostafa A. MOUSA, Mahmoud ELSAMANTY a Ahmed G. RADWAN. Modeling of Soft Pneumatic Actuators with Different Orientation Angles Using Echo State Networks for Irregular Time Series Data. *Micromachines* [online]. 2022, **13**(2), 216 [vid. 2023-05-21]. ISSN 2072666X. Dostupné z: doi:10.3390/MI13020216/S1

[16] 81470264-42f5c800-91ea-11ea-98a2-a8a8d7d96155 [online]. nedatováno [vid. 2023-05-21]. Dostupné z: https://user-images.githubusercontent.com/10376688/81470264-42f5c800-91ea-11ea-98a2-a8a8d7d96155.png

[17] LARGER, Laurent, Antonio BAYLÓN-FUENTES, Romain MARTINENGHI, Vladimir S. UDALTSOV, Yanne K. CHEMBO a Maxime JACQUOT. High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification. *Physical Review X* [online]. 2017, **7**(1), 011015 [vid. 2023-05-21]. ISSN 21603308. Dostupné z: doi:10.1103/PHYSREVX.7.011015/FIGURES/10/MEDIUM

[18] *Reservoir Computing based on Delay-dynamical Systems - YouTube* [online]. [vid. 2023-05-21]. Dostupné z: https://www.youtube.com/watch?v=5AfJaQlPWMU&ab_channel=InstitutodeF%C3%ADsicaInterdisciplinarySistemasComplejos%28IFISC%29

[19] *medium (500×210)* [online]. [vid. 2023-05-21]. Dostupné z: https://journals.aps.org/prx/article/10.1103/PhysRevX.7.011015/figures/1/medium

[20] GAUTHIER, Daniel J., Erik BOLLT, Aaron GRIFFITH a Wendson A.S. BARBOSA. Next generation reservoir computing. *Nature Communications 2021 12:1* [online]. 2021, **12**(1), 1–8 [vid. 2023-05-21]. ISSN 2041-1723. Dostupné z: doi:10.1038/s41467-021-25801-2

[21] *Qubit - Wikipedia* [online]. [vid. 2023-05-21]. Dostupné z: https://en.wikipedia.org/wiki/Qubit

[22] *How To Make A Qubit* [online]. [vid. 2023-05-21]. Dostupné z: https://semiengineering.com/how-to-make-a-qubit/

[23] GIORGINI, Stefano, Bruno JULIA-DIAZ, Sebastiano PILATI a Pere MUJAL. Quantum Reservoir Computing for Speckle Disorder Potentials. *Condensed Matter 2022, Vol. 7, Page 17* [online]. 2022, **7**(1), 17 [vid. 2023-05-21]. ISSN 2410-3896. Dostupné z: doi:10.3390/CONDMAT7010017

[24] *What is Data Reduction? Techniques - Binary Terms* [online]. [vid. 2023-05-21]. Dostupné z: https://binaryterms.com/data-reduction.html

[25] XIE, Haozhe, Jie LI a Hanqing XUE. A survey of dimensionality reduction techniques based on random projection [online]. 2017 [vid. 2023-05-22]. Dostupné z: https://arxiv.org/abs/1706.04371v4

[26] AKANSU, Ali N. a Richard A. HADDAD. Wavelet Transform. *Multiresolution Signal Decomposition* [online]. 2001, 391–442 [vid. 2023-05-21]. Dostupné z: doi:10.1016/B978-012047141-6/50006-9

[27] *Analýza hlavních komponent – Wikipedie* [online]. [vid. 2023-05-21]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Anal%C3%BDza_hlavn%C3%ADch_komponent&oldid=20842743

[28] *What is the basic method of attribute subset selection* [online]. [vid. 2023-05-21]. Dostupné z: https://www.tutorialspoint.com/what-is-the-basic-method-of-attribute-subset-selection

[29] *Attribute Subset Selection in Data Mining - GeeksforGeeks* [online]. [vid. 2023-05-21]. Dostupné z: https://www.geeksforgeeks.org/attribute-subset-selection-in-data-mining/

[30] *AI-Powered Autonomous Machines at Scale | NVIDIA Jetson AGX Xavier* [online]. [vid. 2023-05-21]. Dostupné z: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/

[31] *MongoDB Documentation* [online]. [vid. 2023-05-21]. Dostupné z: https://www.mongodb.com/docs/

[32] *Apache NiFi Documentation* [online]. [vid. 2023-05-21]. Dostupné z: https://nifi.apache.org/docs.html