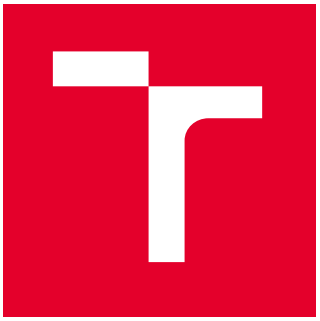



# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

## ÚSTAV AUTOMOBILNÍHO A DOPRAVNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMOTIVE ENGINEERING


# VYUŽITÍ NEURONOVÝCH SÍTÍ PRO ODHAD DYNAMICKÝCH VELIČIN

USE OF NEURAL NETWORKS FOR ESTIMATION OF DYNAMIC VARIABLES

**DIPLOMOVÁ PRÁCE**

MASTER'S THESIS


**AUTOR PRÁCE** — **Bc. Martin Dufek**

AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Lubor Zháňal, Ph.D.**

SUPERVISOR


**BRNO 2024**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

# Zadání diplomové práce

| | |
|---|---|
| Ústav: | Ústav automobilního a dopravního inženýrství |
| Student: | **Bc. Martin Dufek** |
| Studijní program: | Automobilní a dopravní inženýrství |
| Studijní obor: | bez specializace |
| Vedoucí práce: | **Ing. Lubor Zháňal, Ph.D.** |
| Akademický rok: | 2023/24 |

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 diplomové práce:

## Využití neuronových sítí pro odhad dynamických veličin

**Stručná charakteristika problematiky úkolu:**

Moderní prostředky umělé inteligence a strojového učení umožňují klasifikovat a predikovat chování různých systémů či hledat skryté závislosti. Cílem této práce je ověřit využitelnost těchto metod pro oblast vozidlové dynamiky, konkrétně pro získávání specifických veličin, které je obtížné měřit či dopočítávat standardními metodami.

**Cíle diplomové práce:**

Rešerše problematiky strojového učení.
Návrh a sestavení neuronové sítě z vhodných vrstev.
Určení predikovaných veličin a výběr vhodných vstupních kanálů.
Vytvoření ovládacího programu.
Ověření použitelnosti a přesnosti predikovaných veličiny.

**Seznam doporučené literatury:**

GILLESPIE, T. D. Fundamentals of Vehicle Dynamics. Warrendale: Society of Automotive Engineers, 1992. 519 s. ISBN 1-56091-199-9.

VLK, F. Dynamika motorových vozidel. Vyd. 2. Brno: Prof. Ing. František Vlk, DrSc., nakladatelství a vydavatelství, 2006, 432 s. ISBN 80-239-0024-2.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2023/24

V Brně, dne

L. S.

| | |
|---|---|
| prof. Ing. Josef Štětina, Ph.D.<br>ředitel ústavu | doc. Ing. Jiří Hlinka, Ph.D.<br>děkan fakulty |

# ABSTRAKT

Cílem diplomové práce je ověřit použitelnost neuronových sítí k predikci dynamických veličin automobilu. Některé dynamické veličiny vozidel jsou těžko měřitelné, nebo je nutné je dopočítávat, přičemž měření takových veličin může vyjít velmi draze. Pokud by však neuronové sítě dokázaly hodnoty predikovat s přijatelnou chybou, jednalo by se o dostupnější a ekonomičtější metodu. Ověření bylo provedeno vytvořením dvou rekurentních neuronových sítí pro odhad veličin úhlu směrové úchylky a podélných sil na všech kolech automobilu. V práci jsou popsány jednotlivé kroky vytvoření sítí od zpracování vstupních dat až po vyhodnocení predikcí sítí. Výsledky ukazují, že lze využít neuronové sítě k určení dynamických veličin a pro některé účely jimi nahradit drahá měření. V závěru jsou formulovány důležité poznatky získané během vytváření neuronových sítí, které mohou pomoci s vytvářením nových sítí pro odhad dynamických veličin automobilu a je zde nastíněno další možné vylepšení vytvořených neuronových sítí.

## KLÍČOVÁ SLOVA

neuronová síť, rekurentní neuronová síť, odhad dynamických veličin, úhel směrové úchylky, podélné síly na kolech automobilu, Matlab

# ABSTRACT

The aim of the thesis is to verify the applicability of neural networks to predict vehicle dynamic variables. Some vehicle dynamic variables are difficult to measure or need to be calculated, and measuring such quantities can be very expensive. However, If neural networks could predict values with acceptable error, this would be a more affordable and economical method. Verification was performed by creating two recurrent neural networks to estimate the quantities of directional deviation angle and longitudinal forces on all wheels of the car. The paper describes the steps of network creation from processing the input data to evaluating the network predictions. The results show that neural networks can be used to determine dynamic quantities and replace expensive measurements for some purposes. Finally, important insights gained during the creation of neural networks are formulated that can help with the creation of new networks for the estimation of automotive dynamic quantities, and further possible improvements of the created neural networks are outlined.

## KEYWORDS

neural network, recurrent neural network, prediction of dynamics variables, slip angle, longitudinal forces on the vehicle wheels, Matlab

# BIBLIOGRAFICKÁ CITACE

DUFEK, M. *Využití neuronových sítí pro odhad dynamických veličin*. Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automobilního a dopravního inženýrství. Vedoucí diplomové práce Ing. Lubor Zháňal, Ph.D. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/158707.

# ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Lubora Zháňala, Ph.D. a s použitím informačních zdrojů uvedených v seznamu.

V Brně dne 24. května 2024 ……..……..……..

Martin Dufek

# PODĚKOVÁNÍ

Děkuji vedoucímu mé diplomové práce Ing. Luborovi Zháňalovi, Ph.D. za jeho odborné vedení, vstřícnost a přátelský přístup při řešení všech problémů. Dále bych chtěl poděkovat mé přítelkyni a rodině za velkou podporu při studiu.

# OBSAH

# Úvod

Neuronové sítě jsou moderní technologie mající v dnešní době potenciál v mnoha různých oblastech. Jedná se o mocný nástroj v oblasti strojového učení a umělé inteligence. Jsou navrženy pro zpracování dat a provádění složitých úkolů, které by byly pro tradiční algoritmy obtížné či nemožné. Jsou inspirovány mechanismem zpracování informací v lidském mozku a mají schopnost se učit a adaptovat na základě zkušeností, což je jeden z klíčových faktorů, který je odlišuje od „klasického“ programování.

Cílem této práce je využít neuronové sítě k predikci dynamických veličin automobilu. Některé dynamické veličiny vozidel jsou těžko měřitelné, nebo je nutné je dopočítávat. Pokud je však potřeba znát jejich průběh či hodnoty, je nutné provést měření na drahých přístrojích, které tyto těžko měřitelné veličiny dokážou změřit. Takové měření se však může vyšplhat i do velmi vysoké částky. Kdyby však neuronové sítě dokázaly tyto hodnoty predikovat s přijatelnou chybou, jednalo by se o dostupnější a ekonomičtější metodu určení hodnot těžko měřitelných dynamických veličin.

Práce bude zahrnovat celkový proces vytvoření neuronové sítě pro odhad dynamických veličin od volby topologie sítě po samotné otestování její predikce. Konkrétně budou vytvořeny neuronové sítě pro odhad úhlu směrové úchylky a podélných sil na jednotlivých kolech automobilu. Kvůli volbě použití správného druhu sítě budou nejdříve rozebrány topologické druhy neuronových sítí. Následně budou zvoleny jako vstupy základní dynamické veličiny, které jsou dostupně měřitelné s cílem zmenšení úsilí pro získání hodnot těžce měřitelných veličin. Na konkrétních řešených sítích pak bude ukázáno zpracování vstupních dat, které je důležité pro zajištění správného učení sítě. Poté budou vytvořeny architektury neuronových sítí a budou zvoleny hyperparametry a jejich hodnoty. Dále bude ověřena závislost přesnosti predikce vytvořené sítě na jednotlivých vstupních hodnotách pro případné odstranění vstupních veličin, na kterých není síť závislá. A nakonec budou vytvořené sítě otestovány a zhodnoceny. Nejprve na testovacích datech a následně na datasetu, na kterém nebyla síť trénována.

# 1 DYNAMIKA VOZIDEL

Pro lepší pochopení dynamických veličin automobilu je dobré seznámit se s jejich podstatou a obecně dynamikou. Dynamika vozidel je inženýrský obor, zabývající se pohybem vozidla v závislosti na jeho použití. Obor využívá teorie a metody ze strojního a řídicího inženýrství, ale také z konstrukce strojů a věd o lidském chování. Na konci 20. století prošel světový automobilový průmysl mimořádnou proměnou, kdy se do popředí požadavků veřejnosti dostala bezpečnost, ochrana životního prostřední a inteligentní řízení. V důsledku toho se v automobilovém průmyslu staly samozřejmostí moderní technologie, jako jsou řídicí procesory, technologie rozšířené reality, chytré algoritmy, ale i neuronové sítě.

Dynamika vozidel je pro vývoj vozidel klíčová. Dříve se zabývala spíše různými pracovními podmínkami při vnějším buzení, později se ale tento inženýrský obor začal zabývat i řízením, odpružením vozidla a jeho stabilitou za různých jízdních podmínek. Oblastmi, kterým se dostalo v posledních letech velké pozornosti, jsou hlavně komfort a jízdní stabilita.

Části oboru nazývané příčná dynamika a boční dynamika se zabývají zmíněnou jízdní stabilitou. To konkrétněji zahrnuje boční skluz vozidla vyvolaný boční silou pneumatik nebo vychýlení a pohyb vozidla při jeho náklonu. Výzkum jízdní stability v dynamice vozidel postupoval od experimentálních studií k teoretické analýze.

Dalšími částmi oboru jsou podélná dynamika a dynamika vertikální. Tyto části se zabývají jízdou, brzděním a jízdním komfortem. Konkrétněji řeší podélnou sílu pneumatik, která hraje roli při prokluzu, jízdě a brzdění a zároveň zlepšuje jízdní a brzdící účinnosti. Dále řeší vertikální sílu pneumatik, která způsobuje vibrace a náklon vozidla, což ovlivňuje jízdní komfort.

## 1.1 ZÁKLADNÍ ROZMĚRY VOZIDEL

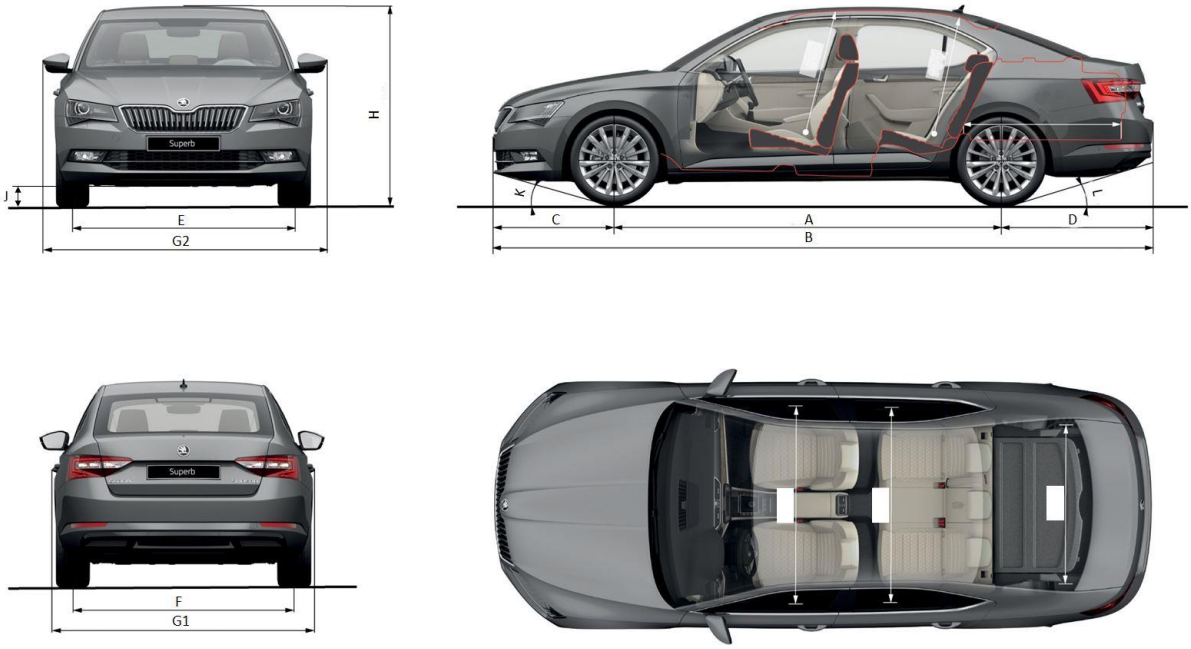
Pro následné výpočty v dalších kapitolách je nutné stanovit a jednoznačně definovat rozměry vozidla. Základní rozměry jsou stanoveny dle normy ČSN 30 0026 [1], která vychází z platné mezinárodní normy ISO 612 [2]. Ke správnému stanovení základních rozměrů vozidel je třeba při měření splnit stanovené předpoklady [3]:

1. Vozidlo je umístěno na vodorovné podložce.
2. Vozidlo je zatíženou přípustnou celkovou nebo pohotovostní hmotností.
3. Vozidlo je v klidu a jeho kola a mechanismus řízení jsou v poloze pro přímý směr.
4. Pneumatiky jsou nahuštěny na tlak odpovídající plnému zatížení automobilu.
5. Vozidlo má standardní příslušenství, výstroj a výbavu.

Základní rozměry vozidla jsou znázorněné na *obr. 1*, kde jednotlivé rozměry představují:

A. Rozvor
B. Celková délka vozu
C. Přední převis
D. Zadní převis
E. Rozchod předních kol
F. Rozchod zadních kol
G. Šířka vozu/šířka vozu se zrcátky

H. Výška vozu
I. Světlá výška vozu
J. Přední nájezdový úhel
K. Zadní nájezdový úhel




*Obr. 1* Základní rozměry vozidel [1]

# 2 PODÉLNÁ DYNAMIKA VOZIDEL

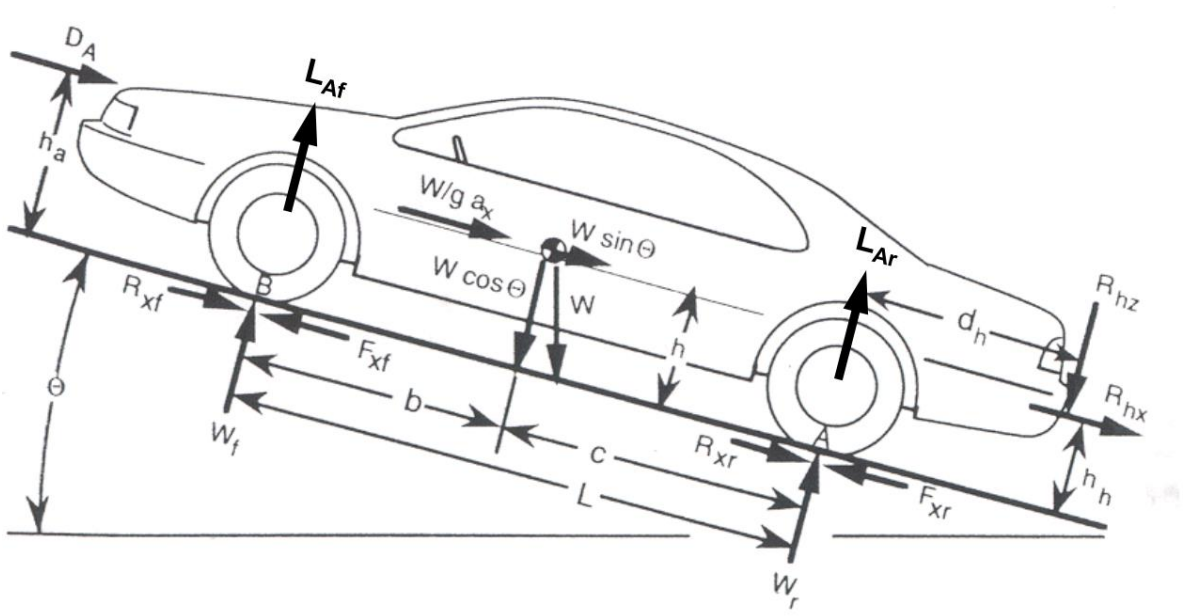
## 2.1 ZATÍŽENÍ NÁPRAV

Momentová rovnice zatížení přední nápravy (1) je odvozena ze základní pohybové rovnice pro pohon viz *obr. 2*.

Momentová rovnice pro zatížení přední nápravy dle [4] zní po odvození následovně:

$$W_f L + L_{Af} L + D_A h_a + \frac{W}{g} a_x h + R_{hx} h_h + R_{hz} d_h + Wh \sin\Theta - \mathrm{W}c \cos\Theta = 0 \qquad (1)$$

kde $W_f$ je zatížení přední nápravy, $L$ je rozvor vozidla, $L_{af}$ je aerodynamický vztlak působící na vozidlo v místě přední nápravy, $D_a$ je aerodynamický odpor působící na karosérii ve výšce $h_a$d, $W/g \cdot a_x$ je setrvačná síla působící v těžišti opačným směrem zrychlení, $h$ je výška těžiště, $R_{hx}$ je podélná síla působící v tažném zařízení ve výšce $h_h$ a vzdálenosti $d_h$ od zadní nápravy, $R_{hz}$ je vertikální síla působící v tažném zařízení ve výšce $h_h$ a vzdálenosti $d_h$ od zadní nápravy, $W$ je tíha vozidla působící v jeho těžišti a $c$ je vzdálenost zadní nápravy od těžiště.




*Obr. 2* Zobrazení působících zatížení na nápravy vozidla [5]

Obdobně zní i momentová rovnice pro zatížení zadní nápravy:

$$-W_r L - L_{Ar} L + D_A h_a + \frac{W}{g} a_x h + R_{hx} h_h + R_{hz}(L + d_h) + Wh \sin\Theta + \mathrm{W}b \cos\Theta = 0 \qquad (2)$$

kde $W_r$ je zatížení zadní nápravy, $L_{ar}$ je aerodynamický vztlak působící na vozidlo v místě zadní nápravy a $b$ je vzdálenost přední nápravy od těžiště.

Z momentových rovnic pro zatížení přední a zadní nápravy lze poté vyjádřit zatížení jednotlivých náprav stojícího, zrychlujícího nebo brzdícího vozidla. Tedy lze zjistit statické zatížení vozidla, nebo podélný přesun zatížení při zrychlení či zpomalení vozidla.

Uvažujeme-li vozidlo bez závěsu, tedy vozidlo, na které nepůsobí podélná a vertikální síla v tažném zařízení a zároveň uvažujeme-li velmi malé stoupání vozovky, po které se vozidlo pohybuje tak, aby bylo možné zanedbat vliv goniometrických funkcí v rovnici, dostáváme rovnici (3) zatížení přední nápravy:

$$W_f = W\frac{c}{L} - W\frac{h}{L}\frac{a_x}{g} - W\frac{h}{L}\Theta + \left(-\mathrm{D_A}\frac{h_a}{L} - L_{Af}\right) \tag{3}$$

a rovnici (4) zatížení zadní nápravy:

$$W_r = W\frac{b}{L} + W\frac{h}{L}\frac{a_x}{g} + W\frac{h}{L}\Theta + \left(D_A\frac{h_a}{L} - L_{Ar}\right) \tag{4}$$

Lze si všimnout, že při zmíněném zjednodušení rovnic zatížení náprav dle (3) a (4) se v rovnicích vyskytují čtyři základní vlivy oddělené pomocí znamének – popořadě: statické zatížení, vliv zrychlení, vliv stoupání a aerodynamické síly. Některé z těchto vlivů budou popsány v dalších kapitolách.

## 2.2 JÍZDNÍ ODPORY

Pohyb automobilů po silnicích a cestách je jedním z nejběžnějších a nejdůležitějších prvků moderního života. V každodenním provozu však automobily musí překonávat různé síly, které brání jejich pohybu a ovlivňují jejich výkonnost. Tyto síly jsou známé pod pojmem jízdní odpory a hrají klíčovou roli ve vývoji vozidla, a to nejen co se týče výkonu, ale i efektivity.

Jízdní odpory zahrnují širokou škálu sil, působících na vozidlo v průběhu jeho pohybu. Každý z těchto jízdních odporů má svůj vlastní vliv na celkovou dynamiku vozidla a spotřebu paliva. Správné porozumění jízdním odporům je klíčové pro maximalizaci efektivity vozidla, minimalizaci spotřeby paliva a zároveň pro zajištění bezpečného a pohodlného pohybu pro posádku. Přesná analýza a modelování jízdních odporů umožňuje navrhnout aerodynamiku vozidla a celkově vyvíjet technologie redukující negativní vlivy jízdních odporů na celkový výkon vozidla.

### 2.2.1 ODPOR VALIVÝ

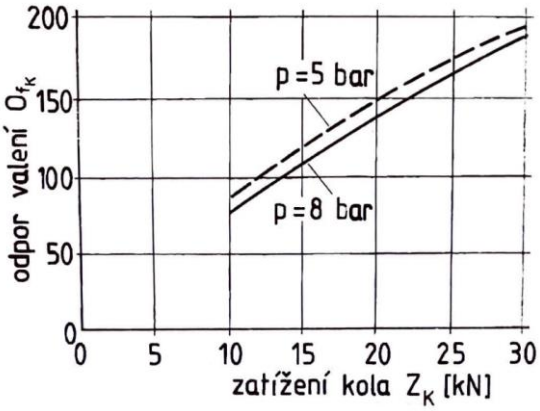
Valivý odpor vzniká deformací pneumatiky a vozovky, ovšem pokud je vozovka dostatečně tuhá, dochází pouze k deformaci pneumatiky. Pneumatika se stýká s vozovkou v ploše nazývané stopa. V přední části stopy dochází ve směru valení ke stlačování pneumatiky do roviny vozovky. V zadní části stopy se pneumatika opět vyrovnává do původního tvaru. Kvůli tepelným ztrátám v pneumatice jsou síly potřebné ke stlačení pneumatiky větší než síly, jimiž působí pneumatika na vozovku. To vede k tomu, že jsou měrné tlaky v přední části pneumatiky větší než v zadní části a výslednice elementárních sil pneumatiky – tzv. radiální reakce vozovky $Z_K$ je předsunuta před svislou osu kola o vzdálenost $e$. Zatížení kola je stejně velké jako reakce

vozovky $Z_K$ a vzniká tedy moment $M_{fK} = Z_K \cdot e$ působící proti smyslu otáčení kola. Dále můžeme reakci $Z_K$ posunout do svislé osy kola, přičemž zavedeme moment $M_{fK}$ působící z vozovky na kolo. Tento moment vyvolá vodorovnou reakci $O_{fK}$, která směruje proti pohybu kola a ve středu kola tak musí z rovnováhy sil působit vodorovná síla $F_{xK} = O_{fK}$. Vodorovnou reakci pak nazýváme valivý odpor kola $O_{fK}$, tedy:

$$O_{fK} = Z_K \frac{e}{r_d} = Z_K f_K \tag{5}$$

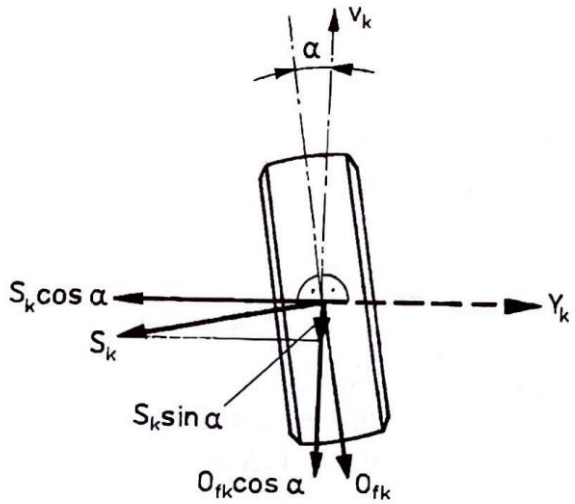
kde $f_K = e/r_d$ je součinitel valivého odporu kola.

Součinitel valivého odporu je závislý především na povrchu vozovky, ale i na vlivu deformace a rychlosti kola. Deformace pneumatiky je závislá na huštění viz *obr. 3*. Při menším tlaku vzduchu v pneumatice dochází k větší deformaci a vzrůstá deformační práce současně se stoupající tlumicí prací, která zvětšuje valivý odpor. Pokud se k tomu přidají ještě vyšší rychlosti, pneumatika nestačí v tak krátkém čase vyrovnávat deformace v přední části stopy a v zadní části stopy tak vzniká menší měrný tlak než při nižší rychlosti. Tím se svislá reakce $Z_K$ posouvá dopředu a součinitel valivého odporu se zvětší. Obecně lze považovat součinitel valivého odporu při nízkých rychlostech (u osobního automobilu do 80 km/h, u nákladního automobilu do 50 km/h) nezávislý na jízdní rychlosti.



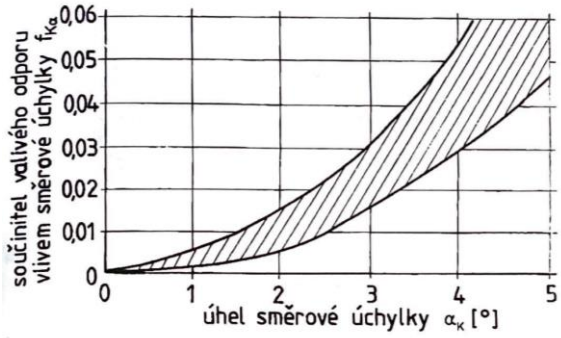

*Obr. 3* Vliv huštění pneumatiky na odpor valení [4]

Kromě valivého odporu působí na vozidlo ještě další odpory velmi podobné tomuto odporu. Patří mezi ně odpor vznikající vlivem sbíhavosti předních kol. Jelikož je však tento odpor velmi malý, obvykle se v praktických výpočtech vůbec neuvažuje. Dále mezi ně patří odpor vznikající vlivem nerovností na vozovkách. Kvůli těmto nerovnostem vznikají v pneumatikách přídavné deformace, které zvyšují jízdní odpor. Další jízdní odpor vzniká při zatáčení vozidla, kdy se kola odvalují se směrovými úchylkami. *Obr. 4* znázorňuje kolo odvalující se pod úhlem směrové úchylky α.




*Obr. 4* Odvalující se kolo pod úhlem směrové úchylky [4]

Ve směru valení kola rychlosti $v_K$ vzniká valivý odpor $O_{fK} \cdot cos\alpha + S_K \cdot sin\alpha$, přičemž $S_K$ je tzv. boční vodící síla kola. Vliv směrové úchylky na součinitel valivého odporu je značně progresivní viz *obr. 5*.




*Obr. 5* Vliv směrové úchylky na součinitel valivého odporu [4]

**Úhel směrové úchylky** je dynamická veličina, která vyjadřuje odchylku směru vozidla od jeho zamýšlené trasy pohybu. Jinak řečeno je to rozdíl mezi směrem, kterým by vozidlo mělo jet, a směrem, kterým skutečně jede. Jak již bylo zmíněno, vzniká v důsledku působení vnějších sil a faktorů, které ovlivňují pohyb a stabilitu vozidla.

### 2.2.2 ODPOR AERODYNAMICKÝ

Při jízdě vozidla se část vzduchu protlačuje prostorem mezi spodní částí vozidla a povrchem vozovky a zbylá část proudí kolem horní části karoserie. Tyto proudnice se za vozidlem ale neuzavírají a vytváří tak víření, které způsobuje vzdušný odpor. Velikost tohoto vzdušného odporu je dána výslednicí normálových tlaků vzduchu na povrch karoserie a třecích sil, které

působí v tečném směru proudění kolem karoserie. Do celkového vzdušného odporu jsou zahrnuty i odpory vznikající vířením a třením vzduchu u otáčejících se kol a odpory vznikající při průchodu vzduchu větracím a chladicím systémem automobilu.

Celkový vzdušný odpor se určí ze vztahu:

$$D_A = c_x \frac{\rho}{2} S_x v_r^2 \tag{6}$$

kde $v_r$ je náporová rychlost proudění vzduchu kolem vozidla, $S_x$ je čelní plocha vozidla, $\rho$ je měrná hmotnost vzduchu a $c_x$ je součinitel vzdušného odporu.

Hodnoty součinitele odporu vzduchu závisí především na tvaru vozidla. Zjišťují se měřením na modelech nebo skutečných vozidlech v aerodynamickém tunelu.

Velká hodnota součinitele odporu vzduchu má za následek velkou spotřebu paliva, a proto je z tohoto hlediska snaha dosáhnout co nejnižších hodnot.

### 2.2.3 ODPOR STOUPÁNÍ

Odpor stoupání je závislý především na tíze vozidla, a tedy na jeho hmotnosti. Působí v těžišti vozidla a je stanoven rovnicí:

$$O_S = \pm G \sin \alpha \tag{7}$$

kde $G$ je celková tíha vozidla a $\alpha$ je úhel, který svírá rovina vozovky s vodorovnou rovinou.

Znaménko se v rovnici určuje podle toho, zda jde o stoupání (plus) nebo o klesání (minus).

Při výpočtech se často odpor stoupání neuvažuje. Pro malé úhly je $\sin \alpha = tg\ \alpha$ a v rovnici (7) lze tedy část s goniometrickou funkcí vynechat. Toto pravidlo lze využít do hodnoty $\alpha = 17$ °, kdy je rozdíl mezi $\sin \alpha$ a $tg\ \alpha$ asi 5 %. Maximální stoupání silnice je 10 až 12 % a na dálnicích asi jen 6 %. Vyšší stoupání mají zpravidla pouze vysokohorské silnice. Odpor stoupání se tedy většinou uvažuje pouze při výpočtu u terénních vozidel.

### 2.2.4 ODPOR ZRYCHLENÍ

Při zrychlování vozidla působí proti směru zrychlení síla setrvačná, kterou nazýváme odpor zrychlení. Odpor zrychlení se skládá z odporů posuvných částí a odporů rotačních částí převodového ústrojí.

Po několika matematických úpravách je dle [4] finální tvar rovnice odporového zrychlení:

$$Oz = \left[1 + \frac{\left(J_m i_C^2 + J_p i_r^2\right)\eta + \sum J_{K_i}}{m r_d^2}\right] m a_x = v m a_x \tag{8}$$

kde $J_m$ je hmotnostní moment setrvačnosti rotujících částí, $i_C$ je celkový převod mezi motorem a hnacími koly, $J_p$ je moment setrvačnosti převodovky, $i_r$ je převod rozvodovky, $\eta$ je celková

mechanická účinnost převodovky a rozvodovky, $J_{Ki}$ je moment setrvačnosti hnacích kol, $m$ je hmotnost posuvných částí, $r_d$ je poloměr kola a $a_x$ je podélné zrychlení.

Celý výraz v hranaté závorce je pak označen jako $\nu$, což je tzv. součinitel vlivu rotačních částí.

Jelikož je celkový převod $i_c$ závislý při stálém převodu rozvodovky na zařazeném převodovém stupni, je také účinek rotačních částí silně závislý na okamžitém převodu. Například při prvním převodovém stupni zvětšují rotační části vozidla potřebnou sílu pro zrychlení u osobních automobilů až o 70 %.

## 2.3 HNACÍ SÍLA

Hnací síla na kolech je síla, která je potřebná pro překonání jízdních odporů vozidla. Z toho vyplývá, že hnací sílu určíme sečtením jednotlivých odporů. Po dosazení za jednotlivé odpory je vyjádření hnací síly dle [4] následovné:

$$F_K = fG + c_x \frac{\rho}{2} S_x v_r^2 + G\left(\tan\alpha + \nu \frac{a_x}{g}\right) \tag{9}$$

Výkon, který musí být na kola přiváděn k překonání jízdních odporů neboli hnací výkon vozidla, je:

$$P_K = F_K v = \frac{M_K}{r_d} v \tag{10}$$

kde $M_K$ je moment na hnacích kolech a v je rychlost vozidla.

Po dosazení jednotlivých odporů do rovnice dostáváme rovnici pro výkon (11), ze které vidíme, že výkon potřebný k překonání většiny odporů roste lineárně s rychlostí jízdy $v$, přičemž jediný rozdíl je u odporu aerodynamického, kde potřebný výkon roste s třetí mocninou rychlosti jízdy.

$$P_K = \left(f + \tan\alpha + \nu \frac{a_x}{g}\right) Gv + c_x \frac{\rho}{2} S_x v^3 \tag{11}$$

## 2.4 TRAKCE

Maximální přenesitelná obvodová síla mezi kolem a vozovkou je podle experimentálních výsledků určena vztahem:

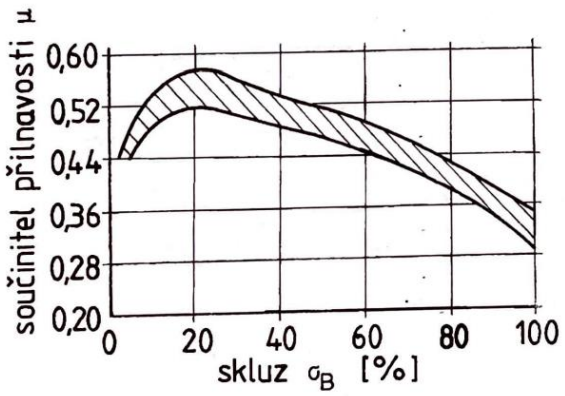
$$H_{K\,max} = \mu_V Z_K \tag{12}$$

kde $\mu_V$ je tzv. součinitel valivé přilnavosti (v podélném směru) a $Z_K$ je tíha hnacích kol.

Je-li hnací moment $M_K$ příliš velký a podélná reakce mezi kolem a vozovkou je větší než $H_{Kmax}$, dochází ve stopě pneumatiky k prokluzování. Dosáhne-li podélný skluz hodnotu $\sigma = 1$, pak je přenášena obvodová síla:

$$H_{K\sigma} = \mu_\sigma Z_K \tag{13}$$

kde $\mu_\sigma$ je tzv. součinitel skluzové přilnavosti.




*Obr. 6* Závislost skluzu na součiniteli přilnavosti [4]

Závislost skluzu na součiniteli přilnavosti a význam součinitelů přilnavosti je názorně vidět v *obr. 6*. Obecně platí, že je součinitel valivé přilnavosti větší než součinitel skluzové přilnavosti. Hodnota součinitele přilnavosti se zjišťuje experimentálně. Je ovlivněna hlavně povrchem vozovky. Pokud je vozovka pokryta vodní vrstvou, musí ji pneumatika narušit, aby došlo ke styku s vozovkou. Je-li rychlost vozidla příliš vysoká, pak pneumatika nestačí odvádět dostatek vody na to, aby došlo ke styku s vozovkou, a dochází k tzv. aquaplaningu.

Pro zajištění co největšího přenosu obvodových sil mezi kolem a vozovkou musí být hnací (popř. brzdné) momenty přiváděné na kolo takové, aby nevzniklo prokluzování kol. Maximální hnací síla na kolech je tedy omezena přilnavostí, tudíž pro celkovou maximální hnací sílu dostáváme dle [4] rovnici:

$$F_{K\,max} = \mu_V G \cos\alpha \tag{14}$$

kde *cos α* je zatížení vozidla kolmé k rovině vozovky.

Maximální točivý moment přiváděný na hnací kola je pak:

$$M_{K\,max} = \mu_V F_{K\,max} r_d \tag{15}$$

Pro ujasnění je dobré zmínit, že přilnavost vozovky sice nepředstavuje meze hnacího motoru, ale vyjadřuje poměry mezi hnacími koly a vozovkou. Tyto znázorněné meze ($M_{K\,max}$) tedy nemá smysl překračovat, protože dochází k prokluzu kol. Spojení mezi kolem a vozovkou je závislé na dosažitelné přilnavosti a pneumatika se odvaluje bez klouzání pouze tehdy, je-li obvodová síla ve vztahu (16). Pro stanovení maximální obvodové síly je nutné vypočítat svislé zatížení kola.

$$H_K \leq \mu_V Z_K \tag{16}$$

# 3 PŘÍČNÁ DYNAMIKA VOZIDEL

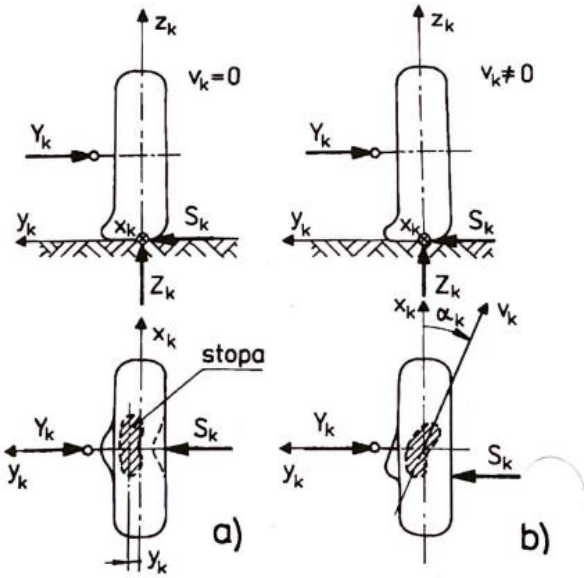
## 3.1 PNEUMATIKY

Pneumatikou rozumíme plášť příp. s duší, ochrannou vložkou, namontovaným na ráfek a naplněný stlačeným vzduchem. Pneumatiky jsou důležité prvky vozidla, které nesou hmotnost vozidla, přenáší hnací a brzdící momenty a boční síly. Dále jsou důležité i v pružící soustavě vozidla a z hlediska zvětšení jízdního pohodlí a bezpečnosti. Pneumatiky vykazují viskoelastické chování, hysterezi a ztráty energie.

Při malé frekvenci zatěžování je hystereze malá, to znamená malé ztráty. S rostoucí frekvencí ztráty rostou. Při velmi vysoké frekvenci zatěžování jsou ztráty natolik velké, že se materiál nestačí vracet zpět do původního tvaru pneumatiky a ta tak získává nový tvar, v kterém v podstatě setrvává a ztvrdne. Na chování pneumatiky má vliv i teplota. Při nízké teplotě je materiál pneumatik tvrdý a křehký. Po překročení teplotního bodu nazývaného „přechod do skelného stavu“ se stává materiál více viskózním. To je způsobeno vyšší mobilitou molekul při zvýšené teplotě. Při zvýšené teplotě se molekuly rychleji vracejí do původního stavu a roste tak frekvence, při které materiál tvrdne.

### 3.1.1 BOČNÍ SÍLA, VRATNÝ MOMENT, SMĚROVÁ ÚCHYLKA

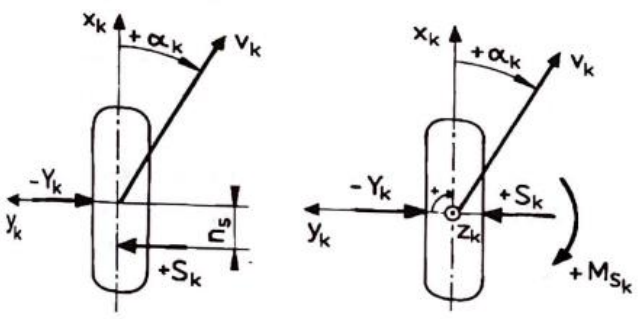
V případě, že na kolo nepůsobí žádná boční síla, je střední rovina kola totožná s podélnou osou stykové plochy pneumatiky s vozovkou. Plocha, kde se pneumatika setkává s vozovkou, se nazývá stopa. Pokud v ose otáčení kola existuje boční síla $Y_K$, vytváří se v stopě vodorovná boční reakce $S_K$. Tato reakce se označuje jako boční vodicí síla kola.




*Obr. 7* Boční a vodicí síla a směrové úchylky:
a) stojící kolo b) valící se kolo [4]

To způsobuje pružnou deformaci pneumatiky v bočním směru a osa stopy se nakloní vzhledem k podélné ose kola v závislosti na velikosti boční síly a boční tuhosti pneumatiky. Když se kolo začne otáčet, jednotlivé elementy na povrchu pneumatiky, které ještě nejsou ve styku s vozovkou, se postupně dostávají do kontaktu s vozovkou, a to způsobuje odchylku osy stopy pod úhlem $\alpha_K$. Vznik směrové úchylky je znázorněn v *obr. 7*.

Otáčející se pneumatika zatížená boční silou se tedy nepohybuje ve směru podélné osy kola. Úhel mezi vektorem rychlosti pohybu kola $v_K$ a podélnou osou kola $x_K$ se nazývá úhel směrové úchylky $\alpha_K$. Odvaluje-li se kolo se směrovou úchylkou, ve stopě pneumatiky vznikají elementární síly vzrůstající směrem k zadnímu konci stopy. Jejich výslednice – boční vodicí síla $S_K$ tedy neleží v ose otáčení kola $y_K$, ale je posunuta směrem dozadu. Rameno boční síly vzhledem k příčné ose kola nazýváme závlek pneumatiky $n_s$. Toto působení boční síly na rameni vytváří moment viz *obr. 8*, který natáčí kolo kolem jeho svislé osy do skutečného směru valení kola a je nazýván vratným momentem kola.




*Obr. 8* Vratný moment, závlek a směrová úchylka na valícím se kole [4]

Při malé směrové úchylce v oblasti $\alpha_K = 0$ až 3 ° lze vyjádřit závislost boční síly na úhlu směrové úchylky:

$$S_K = C_{\alpha K}\alpha_K \tag{17}$$

kde $C_{\alpha K}$ je směrová tuhost pneumatiky. Závlek pneumatiky $n_s$ je pro malý úhel směrové úchylky při stejném zatížení kola přibližně konstantní, takže vratný moment pneumatiky pak můžeme vyjádřit:

$$M_{SK} = S_K n_s = C_{\alpha K} n_s \alpha_K = C_{M\alpha K}\alpha_K \tag{18}$$

Kde $C_{M\alpha K}$ je vratná tuhost pneumatiky.

Na směrové vlastnosti pneumatiky mají vliv různé faktory. Vyšší tlak vzduchu v pneumatice při neměnném svislém zatížení kola snižuje vratnou tuhost a zvyšuje tuhost směrovou. To znamená, že pro stejnou boční sílu bude mít pneumatika s větším huštěním menší směrovou úchylku. Vyšší huštění zmenšuje závlek pneumatiky a snižuje tak vratný moment. Vratná tuhost a tedy i závlek pneumatiky jsou ovlivněny přilnavostí vozovky, přičemž klesají na vozovce

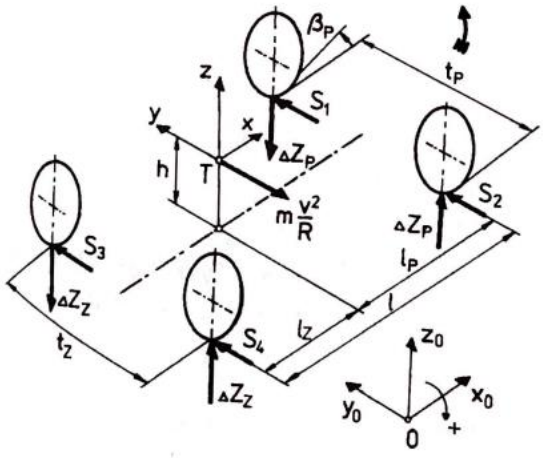
s nízkou přilnavostí. Vliv na vlastnosti pneumatiky má i samotná konstrukce, přičemž rozlišujeme dva základní druhy pneumatiky – radiální a diagonální.

## 3.2 RADIÁLNÍ REAKCE KOL

Odvozené rovnice pro určení radiálních reakcí jsou dle [4]:

$$
\begin{aligned}
Z_1 &= \frac{1}{2} Z_{P_{stat}} - \Delta Z_P = \frac{1}{2} mg \frac{l_Z}{l} - \Delta Z_P \\
Z_2 &= \frac{1}{2} Z_{P_{stat}} + \Delta Z_P = \frac{1}{2} mg \frac{l_Z}{l} + \Delta Z_P \\
Z_3 &= \frac{1}{2} Z_{Z_{stat}} - \Delta Z_Z = \frac{1}{2} mg \frac{l_P}{l} - \Delta Z_Z \\
Z_4 &= \frac{1}{2} Z_{Z_{stat}} + \Delta Z_Z = \frac{1}{2} mg \frac{l_P}{l} + \Delta Z_Z
\end{aligned}
\tag{19}
$$

kde $Z_{Pstat}$ je svislé zatížení předního kola při ustálené poloze, $Z_{Zstat}$ je svislé zatížení zadního kola při ustálené poloze, $\Delta Z_P$ je změna svislého zatížení předního kola, $\Delta Z_Z$ je změna svislého zatížení zadního kola, $m$ je hmotnost vozidla, $g$ je gravitační konstanta, $l$ je rozvor automobilu, $l_Z$ je vzdálenost zadních kol od těžiště automobilu a $l_P$ je vzdálenost předních kol od těžiště automobilu. Jednotlivé vzdálenosti jsou znázorněny v *obr. 9*.




*Obr. 9* Radiální reakce a boční síly na jednotlivých kolech [4]

Změna svislého zatížení předních kol se vypočítá:

$$
\Delta Z_P = m' \frac{v^2}{R} \left[ \frac{l'_Z}{l} \cdot \frac{p_P}{t_P} + \frac{C_P}{C_P + C_Z - m' g' {h_0}'} \cdot \frac{{h_0}'}{t_P} - \frac{m''_P}{m'} \cdot \frac{h''_P}{t_P} \right] \tag{20}
$$

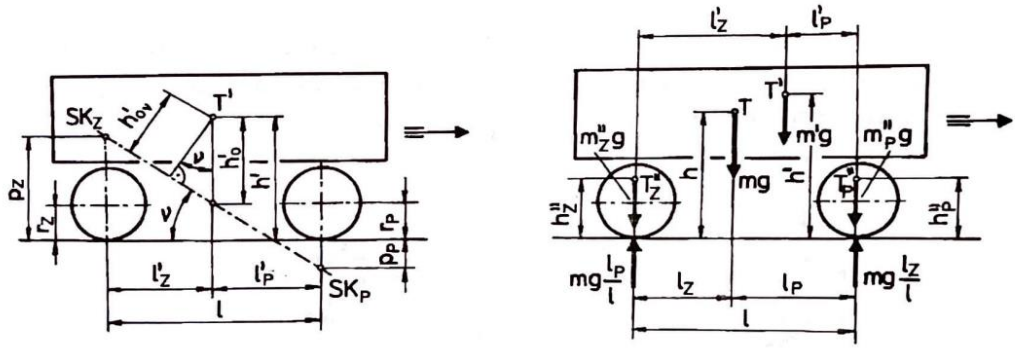
kde $m'$ je hmotnost odpružené části, $m''_P$ je hmotnost přední nápravy, $h_P''$, $h_0'$, $p_P$ a $t_P$ jsou vzdálenosti, které určují polohu těžiště vůči přední části viz *obr. 10*.

Obdobně je tomu i u změny svislého zatížení zadních kol:

$$\Delta Z_Z = m' \frac{v^2}{R}\left[\frac{l'_p}{l}\cdot\frac{p_Z}{t_Z} + \frac{C_P}{C_P + C_Z - m'g'h_0{}'}\cdot\frac{h_0{}'}{t_Z} - \frac{m''_Z}{m'}\cdot\frac{h''_Z}{t_Z}\right] \tag{21}$$

Kde $h_Z''$, $h_0'$, $p_Z$ a $t_Z$ jsou vzdálenosti, které určují polohu těžiště vůči zadní části viz *obr. 10*.

Ze směrových vlastností pneumatiky plyne, že s větším rozdílem reakcí na pravé a levé straně se zvětšuje směrová úchylka nápravy.




*Obr. 10* Geometrie a určení polohy těžiště vozidla [4].

Závislost boční síly pneumatiky na svislém zatížení při stálé směrové úchylce je pro větší zatížení degresivní. Je-li úhel směrové úchylky u obou pneumatik stejný, pak může náprava přenášet největší boční sílu, když jsou obě její kola stejně zatížena v radiálním směru. Čím větší je rozdíl zatížení obou kol, tím je větší úhel směrové úchylky u pneumatik pro danou celkovou boční sílu nápravy.

Vlivem klopení se pravé kolo odlehčí o stejnou hodnotu, o kterou se přitíží kolo levé. Pro stejný úhel směrové úchylky to znamená, že boční vodicí síla na levém kole je větší než na kole pravém.

## 3.3 KLOPENÍ KOL

Vlivem klopení kola vznikne ve stopě pneumatiky boční síla. Ta je určena výrazem:

$$S_K = C_{\alpha K}\alpha_K - C_\xi \xi \tag{22}$$

kde $C_\xi$ je klopná tuhost pneumatiky a $\xi$ je úhel naklopení kola.

Naklápěním přední nebo zadních kol lze ovlivňovat vznik boční vodicí síly. Ta je zároveň závislá na svislém zatížení kola. Tímto efektem jde tedy ovlivňovat přetáčivost nebo nedotáčivost vozidla.

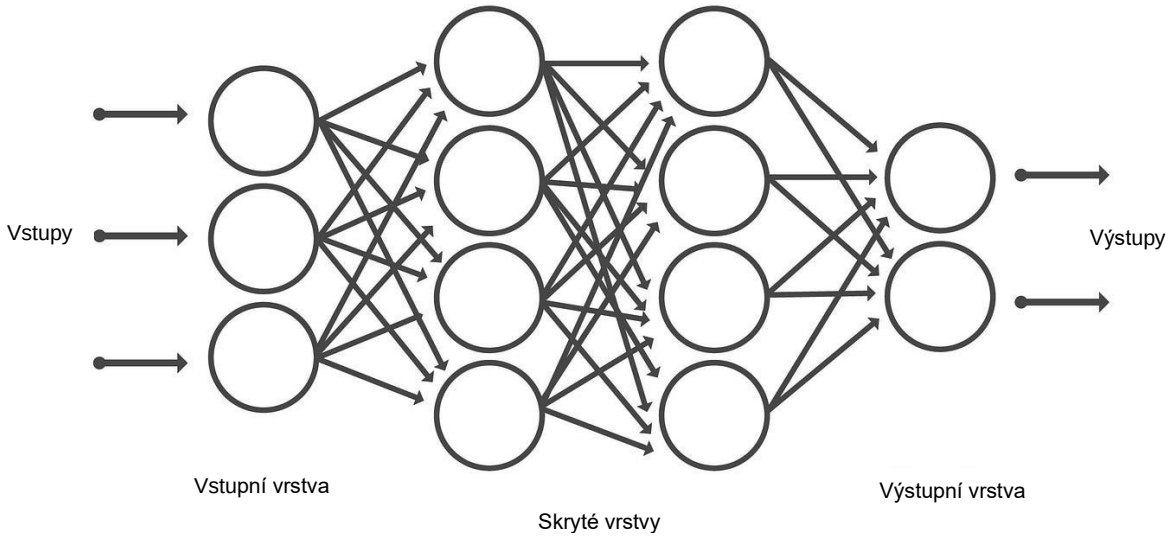
# 4 NEURONOVÉ SÍTĚ

Neuronové sítě, často nazývané z angličtiny umělými neuronovými sítěmi (Artifical Neural Networks) nebo hlubokými neuronovými sítěmi (Deep Neural Networks) představují zajímavou oblast umělé inteligence a strojového učení.

Mají různé topologie a architektury závislé na oblasti využití. Lze je využít například na zpracování obrazu, zpracování a porozumění řeči, zpracování jazyka, předpovídání, diagnostiky a další. Čím více má síť vrstev, tím více je síť hlubší a je schopná se naučit více abstraktních funkcí. Je to taky důvod, proč je oblast hlubokého učení, využívajícího více vrstev neuronových sítí, čím dál tím více populární. Velmi důležité je však správné předzpracování trénovacích dat tak, aby bylo zajištěno správné učení sítě.

Neuronové sítě mají obrovský potenciál a stávají se stále důležitějšími v různých odvětvích, včetně medicíny, průmyslu, financí a celkově v oblasti vědeckého výzkumu. Jejich schopnost zpracovávat složité vzory a učit se z dat je klíčem k řešení komplexních problémů a dosahování inovací.

Neuronová síť, inspirovaná (ale fungující jinak) biologickými nervovými systémy, kombinuje několik vrstev zpracování pomocí jednoduchých paralelně pracujících prvků. Základní schéma typické architektury neuronové sítě je na *obr. 11*.




*Obr. 11* Typická architektura neuronové sítě [6]

Síť se skládá ze vstupní vrstvy, jedné nebo více vrstev skrytých a z vrstvy výstupní. V každé vrstvě je pak několik neuronů, které používají jako vstupy výstupy neuronů v předchozí vrstvě, všechny neurony jsou také navzájem propojeny. Každému neuronu je pak obvykle přiřazena nějaká váha, která se upravuje v průběhu procesu učení. Snížení nebo zvýšení dané váhy mění sílu signálu neuronu s cílem minimalizovat chybu předpovědi.

## 4.1 KLASIFIKACE A REGRESE

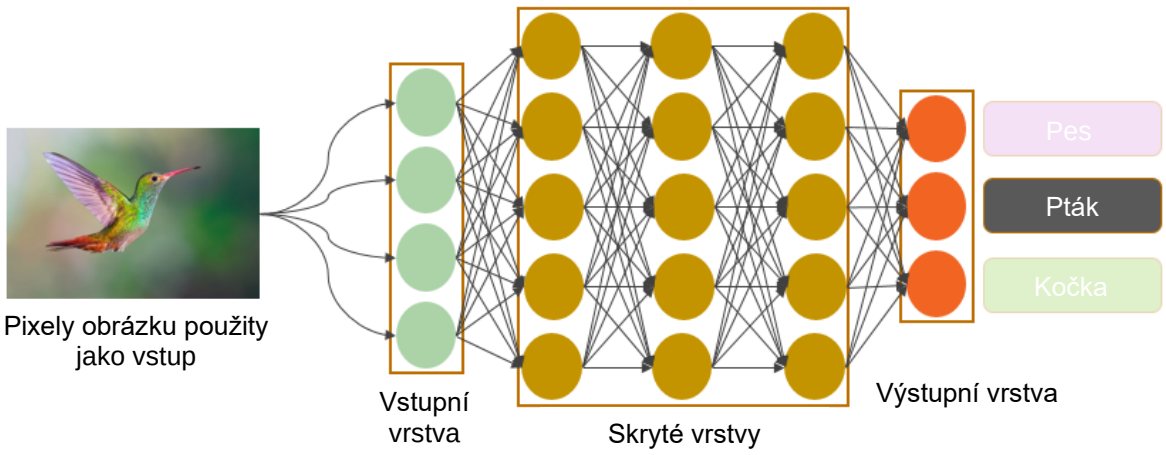
V oblasti zpracování dat a strojového učení jsou základními stavebními kameny prediktivní analýzy dvě úlohy – klasifikace a regrese. Obě tyto techniky hrají klíčovou roli při řešení různých problémů, protože nám umožňují provádět predikce a rozhodovat se na základě souboru dat.

Regrese je typ učení, který se zabývá předpovídáním spojitých hodnot nebo číselných výsledků. Zahrnuje nalezení matematického vztahu mezi vstupními a výstupními prvky a cílovou proměnnou, což nám umožňuje provádět předpovědi v daném rozsahu [7]. Mezi běžné příklady použití regrese patří předpovídání cen domů na základě vlastností, jako jsou počet místností, lokalit a plocha, nebo předpovídání cen akcií na základě dat z minulosti a ekonomických ukazatelů.

Mezi hlavní vlastnosti regrese patří:

- Spojité výstupy – cílem regrese je předpovídat spojitý číselný výstup.
- Metriky hodnocení – regresní model se hodnotí pomocí metrik, jako jsou střední kvadratická chyba (MSE), odmocnina ze střední kvadratické chyby (RMSE) nebo střední absolutní chyba (MAE), které určují přesnost předpovědi.
- Algoritmus – regresní algoritmy jsou různé, přičemž nejjednodušší a nejčastěji používaná je lineární regrese. Další algoritmy jsou pak složitější, ale flexibilnější pro různé typy dat.

Klasifikace se na druhou stranu zaměřuje na kategorizaci dat do předem definovaných tříd nebo skupin. Tento druh úlohy se tedy používá na cílové proměnné, které lze zařadit do různých požadovaných tříd [7]. Klasifikační problémy jsou například určení, o které zvíře na obrázku se jedná viz *obr. 12*, nebo rozpoznání ručně psaných písmen či čísel.




*Obr. 12* Typické schéma klasifikační konvoluční neuronové sítě [8]

Hlavní vlastnosti klasifikace jsou:

- Diskrétní výstupy – klasifikace přiřazuje body do určitých kategorií nebo tříd, často reprezentovány celými čísly. Tyto třídy mohou být binární nebo i vícetřídní.

- Metriky hodnocení – mezi běžné metriky hodnocení klasifikace patří přesnost odhadu, preciznost odhadu a odvolání.
- Algoritmus – klasifikační algoritmy zahrnují širokou škálu metod, včetně logistické regrese, rozhodovacích stromů, náhodných lesů atd. Výběr závisí na datech a daném problému.

# 5 DRUHY NEURONOVÝCH SÍTÍ

V oblasti umělé inteligence a strojového učení zaznamenala neuronová síť významný vývoj od svých počátků v roce 1958, kdy Frank Rosenblatt představil síť s názvem Perceptron. Tato jednoduchá struktura se skládala z jediného neuronu a lineárního regresního modelu se sigmoidní aktivační funkcí. Od té doby však prošly neuronové sítě velkým vývojem a postupně se zdokonalovaly, čímž se dostaly od mělkých neuronových sítí (shallow neural networks) až k dnešním komplexním hlubokým sítím, schopným obsahovat až stovky vrstev.

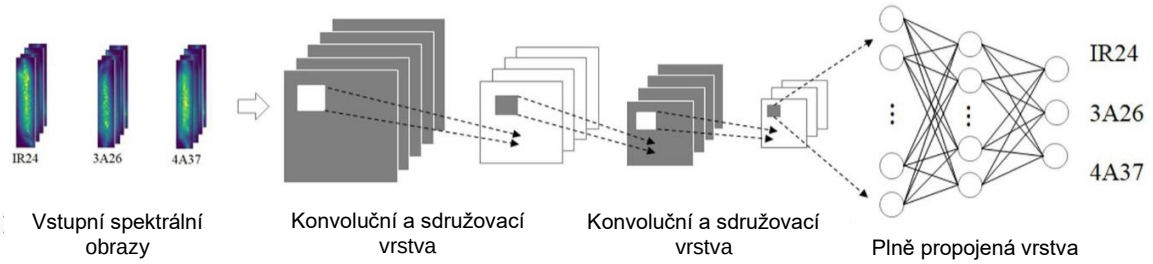
V rámci této problematiky byly vytvořeny tři základní typy neuronových sítí, které budou v této práci rozebrány a porovnány. Tyto typy jsou klíčové komponenty moderního strojového učení a jejich porozumění představuje klíčový krok k úspěšnému využití neuronových sítí v různých oblastech.

## 5.1 KONVOLUČNÍ NEURONOVÁ SÍŤ

Konvoluční neuronová síť (CNN) představuje významný model hlubokého učení, který byl speciálně navržen pro manipulaci s daty uspořádanými do maticové struktury. Tato síť je využívána zejména k úlohám klasifikace obrázků, počítačového vidění a zpracování přirozeného jazyka [9]. Dokáže automaticky extrahovat vlastnosti obrazu. Má zřejmé výhody při rozpoznávání obrazu a lze ji použít pro trénování rozsáhlých dat. Na *obr. 12* je typické schéma konvoluční neuronové sítě.

### 5.1.1 STRUKTURA KONVOLUČNÍ SÍTĚ

Strukturu konvoluční neuronové sítě lze rozdělit na vrstvu konvoluční, sdružovací a plně propojenou dle různých způsobů výpočtu. Ukázka struktury konvoluční sítě je na *obr. 13*. Mapa prvků je generována extrakcí lokálních prvků vstupních dat pomocí konvoluční vrstvy. Dále se zmenší rozměr mapy prvků pomocí sdružovací vrstvy, a nakonec se zpracovaná mapa prvků vloží do plně propojené vrstvy, přičemž výstupem je výsledek dle různých úloh.




*Obr. 13* Struktura konvoluční neuronové sítě [10]

Konvoluční vrstva shromažďuje lokální prvky trénovacích dat pomocí konvolučních operací. Vstupní data a váhové parametry se po konvolučních operacích spojí s hodnotou posunu a výsledek je vstup do aktivační funkce. Velikost finálního grafu parametrů souvisí s nastavením jednotlivých parametrů konvoluční neuronové sítě. Vzorec pro výpočet konvoluce je:

$$x_{ij}^{(l)} = f\left(\sum_{m=0}^{k}\sum_{n=0}^{k} x_{i=m,j+n}^{(l-1)} w_{mn}^{(l)} + b^{(l)}\right) \tag{23}$$

Kde $x$ je pixel obrazu, $k$ je délka nebo šířka konvolučního jádra, $w$ je váhový vektor, $b$ je zkreslení a $f$ je aktivační funkce.

Vstupem do konvoluční neuronové sítě je obraz jako datová matice. Konvoluční jádro je pak v podstatě také lokální váhovou maticí. Operace konvoluce spočívá v tom, že se konvoluční jádro posouvá po vstupních datech v určitém počtu kroků, přičemž se toto pole často překrývá, násobí se a sčítá prvek po prvku pro každou pozici, aby se získala nová 2D matice prvků [10]. Operace konvoluce umí účinně extrahovat lokální prostorové informace ve vstupních datech. Z konvoluční vrstvy vychází stejně velký výstup, jaký byl vstup.

Sdružovací vrstva funguje na podobném principu. Podobně jako u konvolučních operací se sdružování realizuje pohybem posuvného okna na grafu prvků [11], ovšem obvykle zde nedochází k překrývání pole. Podle velikosti kroku se při každém pohybu na mapě prvků získá reprezentativní hodnota dané oblasti. Po provedení operace sdružování se celková velikost mapy prvků zmenší. Výpočetní vzorec pro operace sdružování je dle [11] následující:

$$x^t = R(\beta^t \cdot P(x^{t-1}) + b^t \tag{24}$$

Kde $R$ je aktivační funkce, $P$ je sdružovací funkce, $\beta$ je váhový koeficient a $b$ je zkreslení.

Sdružovací operace rozdělí vstupní mapovací matici prvků na nepřekrývající se oblasti a poté provede sdružení každé oblasti výpočtem průměru nebo výběrem maximální hodnoty. Operace sdružování agreguje nebo počítá hodnoty v každém okně vytvořeném posouváním posuvného okna v dané matici a vyvede je do další vrstvy jako vstupní data. Sdružování obvykle následuje po operaci konvoluce a zmenšuje tak rozměry matice prvků na výstupu z konvoluční vrstvy, díky čemu jsou účinně zachovány důležité rysové informace i přes provedené zmenšení matice.

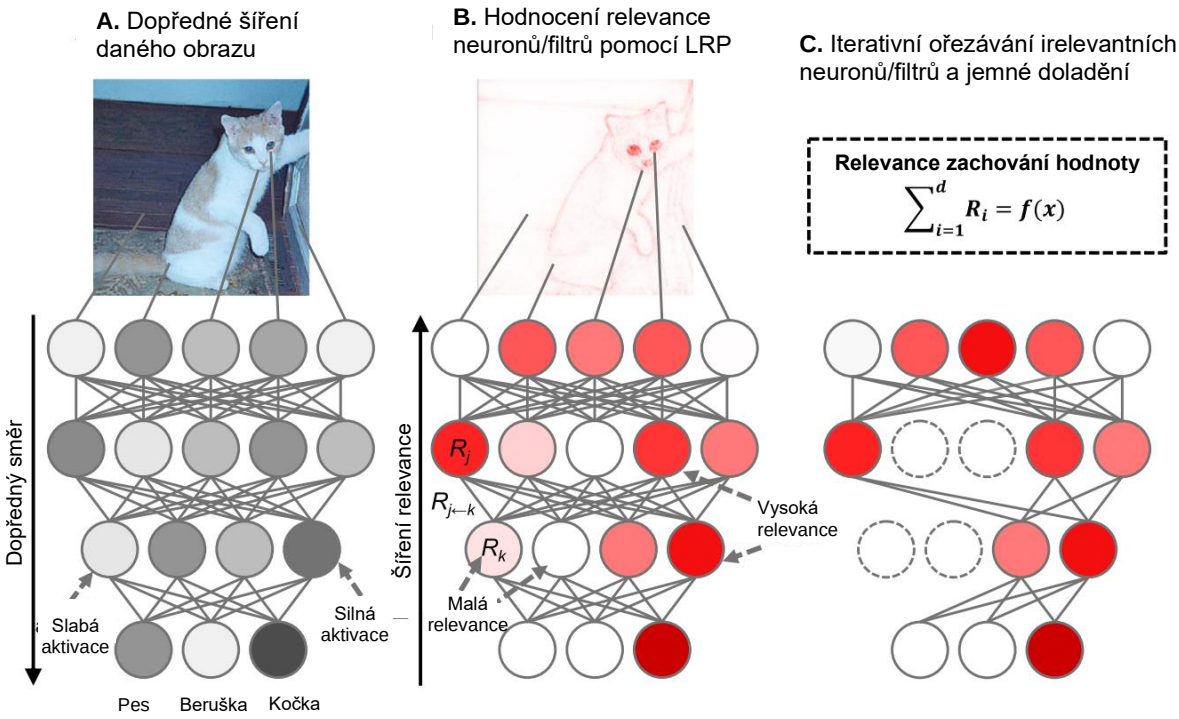
Plně propojená vrstva je následně umístěna v poslední CNN. Režim plného propojení neuronů se využívá k mapování dvourozměrné mapy prvků do jednorozměrného vektoru, a nakonec k mapování do prostoru vzorků podle různých druhů úloh. Předchozí vrstvy – konvoluční a sdružovací mapují vstupní data do reprezentativního prostoru prvků, následně je tento reprezentativní prostor prvků mapován plně propojenou vrstvou do prostoru pro označení vzorků, aby bylo možno dosáhnout klasifikace nebo regrese.

### 5.1.2 OPATŘENÍ PRO ZMENŠENÍ KONVOLUČNÍ NEURONOVÉ SÍTĚ

Klasické CNN mají často problém s příliš mnoha parametry. Aby bylo možné model sítě vylepšit a zlepšil se tak výkon a přesnost klasifikace, je zapotřebí mnoho vstupních vzorků. Ovšem v praktických technických aplikacích je obtížné získat velký počet vstupních vzorků, které by podpořily přesnost klasifikace. Poptávka po nasazení konvolučních neuronových sítí do systémů používaných v běžném každodenním životě neustále roste a je zapotřebí snížit náročnost architektury CNN. Za předpokladu, že se nesníží výkonnost modelu, by se měl co nejvíce snížit objem a náročnost architektury a výpočetního výkonu, aby se tak dosáhlo co nejlepšího poměru mezi výkonem a využitím. V současné době se tato náročnost architektury snižuje především prostřednictvím komprese modelu a úpravy konvolučních operací.

Nejběžnější metody komprese modelu jsou ořezávání sítě, kvantizace parametrů, aproximace s nízkou hodnotou a destilace znalostí.

Ořezávání sítě se hojně využívá jako klasická technika v oblasti komprese modelů při zpracování různých algoritmů. Jedná se o důležitou techniku, která dokáže snížit složitost sítě a zabránit jejímu nadměrnému přizpůsobení [12]. Ořezávání má obvykle 3 základní kroky – trénování spojení a měření důležitosti neuronů, odstranění nedůležitých neuronů a přeučení a jemné vyladění sítě.




*Obr. 14* Ořezávání sítě – v tomto případě obrazu kočky [12]

Kvantizace parametrů je operace, která nahrazuje všechny původní parametry novými dílčími parametry, což výrazně snižuje náklady na velikost úložiště. Kvantizací, jakožto běžnou technikou zpětné komprese, lze dosáhnout velkého zmenšení objemu modelu při malých ztrátách výkonu sítě [13]. Nevýhodou kvantizace je, že kvantizovaná sít je již pevně daná a lze ji jen obtížně měnit. Zobecnění tohoto způsobu je špatné a vede k vysokým nákladům na údržbu. Jedním z nejjednodušších kvantizačních algoritmů je skalární kvantizace.

Aproximace s nízkou hodnotou rozkládá obrovskou hustou váhovou matici na několik malých matic, které dohromady přibližně rekonstruují původní váhovou matici [14]. Tato operace dosahuje snížení objemu paměti, a i potřebného výpočetního výkonu. Ve skutečné implementaci konvoluční operace je operace konvoluce ukončena násobením matic. Váhová matice však obecně bývá hustá a velká, což přináší nutnost velkého výpočetního výkonu. Hlavní myšlenka aproximace s nízkou hodnotou je tedy taková, že pokud lze hustou matici přibližně rekonstruovat pomocí několika menších matic, pak lze efektivně snížit náklady na výpočet a ukládání.

Destilace znalostí je metoda přenosu učení s cílem převedení znalostí získaných z komplexního síťového rámce do malého kompaktního modelu pomocí určitých přístupů tak, aby i malé

schéma mohlo získat podobné schopnosti jako velký komplexní model. V rámci této metody hrají rozhodující roli dva základní prvky: první – co je to „znalost“ a jak znalosti v modelu extrahovat a druhý – jak provést destilaci, tedy jak dokončit proces předávání znalostí.

Ačkoliv komprese modelu zaujímá v procesu odlehčení konvoluční neuronové sítě významnou roli, její proces je příliš složitý a obvykle vyžaduje podrobné opakované trénování, aby bylo dosaženo podobného výkonu jako u původního modelu. Proto někteří výzkumníci začínají přímo navrhovat odlehčené verze konvolučních neuronových sítí s cílem kontrolovat počet parametrů a výpočet pomocí skupinové konvoluce, hluboké oddělitelné konvoluce, hluboké konvoluce a bodové konvoluce.

Skupinová konvoluce rozdělí rozměr funkčních kanálů na několik stejných částí, které samostatně projdou konvolucí a následně výsledky shromáždí. Myšlenka skupinové konvoluce se nejvíce využívá při návrhu sítí. Kromě toho, že snižuje počet parametrů, ji lze také považovat za strukturovaný zjednodušený přístup, který je ekvivalentní regulačnímu způsobu. S rostoucím počtem filtrů se snižují parametry modelu a rámec se stává efektivnější. Vzhledem k tomu, že konvoluce je rozdělena na více cest a každá cesta může být zpracována samostatně prostřednictvím daného GPU, lze architekturu sítě trénovat na několika GPU současně, čímž se výrazně zvýší rychlost trénování.

Hluboká oddělitelná konvoluce má jistou přednost, a to takovou, že čím více atributů je potřeba extrahovat, tím více parametrů lze ušetřit, čímž se sníží objem výpočtů. Hluboká oddělitelná konvoluce je vlastně druh rozkládací konvoluční operace, která zahrnuje prostorovou dimenzi, ale zabývá se také dimenzí hloubkovou. Lze ji rozdělit na dvě menší operace – hlubokou konvoluci a bodovou konvoluci.

Hluboká konvoluce je druh paketové konvoluce, ve které je počet paketů roven počtu kanálů funkce a konvoluční jádro je rovné odpovídajícím kanálům jedna ku jedné. Hloubka výstupní mapy prvků je tak stejná jako hloubka vstupní mapy. V případě vícekanálových grafů charakteristik z předchozí vrstvy se nejprve všechny rozdělí na grafy charakteristik jednoho kanálu a provede se na nich jednokanálová konvoluce. Následně jsou poskládány dohromady. Na rozdíl od standardních konvolučních činností rozdělí hluboká konvoluce konvoluční jádro na několik kanálů a provádí konvoluční operace na každém kanálu, aniž by se měnila hloubka vstupního obrazu. Tímto způsobem se získá výstupní graf vlastností se stejným počtem kanálů jako vstupní graf vlastností.

Hluboká konvoluce upravuje pouze rozměr grafu charakteristik z předchozí vrstvy, zatímco počet kanálů se nemění. To vyžaduje další úpravu – bodovou konvoluci. Jedná se o konvoluci o velikosti $1 \times 1$. Jelikož hluboká konvoluce nespojuje informace mezi kanály, je třeba použít ji společně s konvolucí bodovou. To umožní zvýšit nebo snížit dimenzi grafů charakteristik. Operace bodové konvoluce je podobná klasické konvoluci, která spočívá ve vytvoření nové mapy příznaků váženým sčítáním mapy charakteristik předchozího stupně [15]. Každý filtr exportuje jednu mapu vlastností, tudíž několik kanálů vyžaduje více filtrů.

### 5.1.3 KONVOLUČNÍ SÍŤ – SHRNUTÍ

Jak již bylo zmíněno, pro rozpoznávání obrazu, klasifikaci obrazu a aplikace počítačového vidění jsou konvoluční sítě obzvláště užitečné, protože poskytují velmi přesné výsledky, a to hlavně tehdy, jedná-li se o velké množství dat. Konvoluční sítě se také učí poznat

charakteristiky daného objektu v postupných iteracích tím, jak data procházejí mnoha jednotlivými vrstvami. Konvoluční sítě tak eliminují potřebu ruční extrakce příznaků. Lze je také přeučit na nové úlohy rozpoznávání anebo je navázat na již existující sítě. Tyto výhody tedy otevírají nové možnosti využití konvolučních sítí pro reálné aplikace bez zvyšování výpočetních nákladů. Konvoluční sítě jsou pak obecně výpočetně efektivnější než klasické neuronové sítě [16] a mohou tedy běžet na jakémkoliv zařízení, včetně chytrých telefonů.

## 5.2 REKURENTNÍ NEURONOVÁ SÍŤ

Pokud jsou záznamy datového souboru uloženy postupně a existuje mezi nimi závislost, považuje se tento datový soubor za sekvenční. Dobrým příkladem jsou časové řady dat, např.: sekvence genů, údaje o počasí, srdeční frekvence, ceny akcií a další. Vyskytují se však typicky i v inženýrských problémech, jako je předpověď různých veličin, životností či poruchovosti a hrají tedy důležitou roli v oborech inženýrství spolehlivosti, monitorování systémů a prognostice. Přestože lze takový druh dat zpracovávat různými metodami, jako jsou modely autoregresního integrovaného klouzavého průměru nebo exponenciálního vyhlazování, v posledních desetiletích vzrostla popularita neuronových sítí [17]. Přesněji řečeno, rekurentní neuronové sítě poskytly vynikající výkon při aplikaci na sekvenční data časových řad. Existuje mnoho různých variant rekurentních sítí, od Long Short Term Memory (LSTM) po Gated Recurrent Unit (GRU) a jsou zodpovědné za mnoho nástrojů a softwaru, které jsou využívány v každodenním životě – např.: rozpoznávání řeči, strojový překlad, analýza sentimentu, hudba a další. O využití RNN se stále více zajímá i strojírenský a stavebnický průmysl. Jejich úspěch v tolika různých oblastech použití spočívá hlavně v jejich schopnosti ukládání užitečných informací z minulosti a následně jejich využití k předpovědím budoucích stavů.

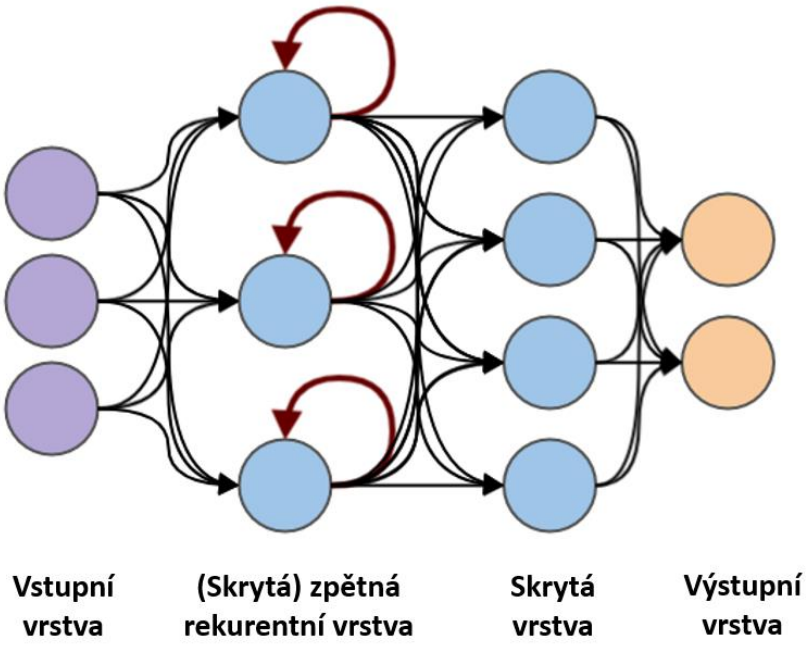
Rekurentní neuronové sítě mají ovšem i určité problémy. Většinou související s použitím algoritmu zpětného šíření k trénování vah neuronové sítě, konkrétně s mizejícími a explodujícími gradienty [18]. Tyto problémy se sice netýkají jenom RNN, ovšem skutečnost, že se gradient ztrátové funkce zpětně šíří v mnoha časových krocích, způsobuje, že tento typ neuronové sítě je k nim náchylnější. Pro vyřešení tohoto problému byla navržena různá řešení, přičemž nejlepší výsledky poskytují typy sítí LSTM a GRU. Další obvyklou nevýhodou neuronových sítí je špatná výkonnost při nedostatku či nevyváženosti vstupních dat, což je v inženýrství běžná situace.

Oba problémy jsou dále řešeny hybridními metodami, které do algoritmů založených na datech zavádějí fyzikální modely. Existují různé RNN, které jsou kombinovány s fyzikou a vytvořily tak sítě hybridní. V závislosti na povaze fyzikálních zákonů jsou tyto RNN kombinovány různým způsobem.

### 5.2.1 ZÁKLADNÍ PRINCIPY REKURENTNÍ NEURONOVÉ SÍTĚ

Koncept RNN se zrodil v 80. letech 19. století z důvodu potřeby zpracovávat neuronovými sítěmi sekvenční data. RNN se totiž vypořádaly se dvěma hlavními problémy, které jsou s takovým typem dat spojeny: vstupní informace různých délek a skutečnost, že předchozí vstupy mohou ovlivňovat budoucí vstupy i výstupy. To vše je uskutečněno pomocí sdílení parametrů, kdy se v každém časovém kroku neuronové sítě rekurzivně používají stejné váhy. Tímto způsobem se pak ukládají informace o bezprostřední minulosti a přidávají se

k přítomnosti, aby bylo možné následně předpovídat budoucnost. To znamená, že rekurentní neuron má tedy dva vstupy – jeden pro aktuální vstupní data a druhý pro výstup z předchozího kroku. Základní princip rekurentní neuronové sítě je znázorněn na *obr. 15*.




*Obr. 15* Základní schéma rekurentní neuronové sítě [19]

Sdílení vah je důležité zejména tehdy, když se určitá informace může vyskytovat v dané sekvenci na více místech. Například věta „V roce 2022 jsem byl v Barceloně“ oproti větě „V Barceloně jsem byl v roce 2022“ – pokud bychom požádali model neuronové sítě, aby přečetl každou větu zvlášť a extrahoval rok, ve kterém byl vypravěč v Barceloně, požadavek je, aby poznal rok 2022 jako relevantní informaci, ať už se objeví ve třetím nebo sedmém slově věty. Pokud by se jednalo o natrénovanou plně propojenou dopřednou síť, která zpracovává věty pevné délky, síť by měla samostatné parametry pro každou vstupní funkci, takže by se musela naučit všechna pravidla jazyka zvlášť pro všechna pořadí slov ve větě [20]. Ovšem právě díky sdílení vah RNN může porovnat jednotlivá pořadí ve větě mezi sebou.

Související myšlenkou je použití konvoluční posloupnosti v 1D. Tento konvoluční přístup je základem časově zpožděných neuronových sítí. Operace konvoluce umožňuje síti sdílet parametry napříč časem, ale je pouze povrchní. Výstupem konvoluce je posloupnost, kde každý člen výstupu je funkcí malého počtu sousedních členů vstupu. Myšlenka sdílení parametrů se pak projevuje použitím stejného konvolučního jádra v každém časovém kroku. Ovšem RNN sdílejí parametry jiným způsobem. Každý člen výstupu je funkcí předchozích členů vstupů. Každý člen výstupu je tedy vytvářen pomocí stejného aktualizačního pravidla, které bylo použito na předchozí výstupy. Tato rekurentní formulace vede ke sdílení parametrů prostřednictvím velmi hlubokého výpočetního grafu. RNN je možné použít i ve 2D napříč prostorovými daty, jako jsou obrázky. Dokonce při použití na data zahrnující čas může mít síť spojení, která jdou zpět v čase, za předpokladu, že celá sekvence je pozorována předtím, než je poskytnuta síti.

Existuje mnoho různých typů RNN. V závislosti na vztahu vstup – výstup v datech existuje několik vzorů:

- jeden vstup – jeden výstup,
- jeden vstup – více výstupů,
- více vstupů – jeden výstup,
- více vstupů – více výstupů.

Dobrými příklady vzorů více vstupů – jeden výstup jsou analýza sentimentu nebo klasifikace textu. Příklad více vstupů – více výstupů je např.: strojový překlad. Počet proměnných také určuje, zda je model jedno variantní nebo více variantní. Původní koncepce rekurentní sítě se navíc vyvinula do složitějších a sofistikovanějších algoritmů, jako je obousměrná rekurentní neuronová síť, GRU nebo LSTM.

Navzdory tomu, že se RNN těší velkému úspěchu v různých aplikacích, mají i svá negativa. Většinou se spoléhají na algoritmus zpětného šíření, který se ovšem v rekurentních sítích stává složitějším a je tak zdrojem problémů, jako jsou již zmíněné mizející a explodující gradienty. Jakožto každá neuronová síť si RNN nevedou moc dobře při provádění předpovědi mimo oblast trénovacích dat. Ve skutečnosti se tento problém ještě zhoršuje při jednorozměrném vícekrokovém předpovídání, kdy se předpovídaná hodnota aktuálního časového kroku používá k určení hodnoty dalšího časového kroku. Tato situace je běžná v oblasti inženýrství – v oblasti prognostiky a řízení stavu, kde se aktualizované informace o stavu konstrukce rekurzivně používají k prognóze budoucího stavu způsobilosti systému.

### 5.2.2 GATED RECURRENT UNIT (GRU) NEURONOVÁ SÍŤ

Jedná se o typ architektury rekurentní neuronové sítě, podobný architektuře LSTM. Stejně jako LSTM je GRU navržena k vytvoření neuronové sítě se zaměřením na sekvenční data tím, že umožňuje selektivní zapamatování nebo zapomenutí informací v průběhu času [21]. GRU má však jednodušší architekturu než LSTM s menším počtem parametrů, což usnadňuje trénování sítě a zvýšení výpočetní efektivity.

Hlavní rozdíl mezi GRU a LSTM spočívá ve způsobu, jakým zpracovávají stav paměťových buněk. V LSTM je stav paměťové buňky udržován odděleně od skrytého stavu a je aktualizován pomocí tří bran – vstupní brána, výstupní brána a brána zapomínání. V GRU je stav paměťové buňky nahrazen průběžným aktivačním vektorem, který je aktualizován pomocí dvou bran – resetovací brána a brána aktualizace.

Resetovací brána určuje, kolik se má zapomenout z předchozího skrytého stavu, zatímco brána aktualizace určuje, kolik se má začlenit z průběžného aktivačního vektoru do nového skrytého stavu. GRU je pak celkově oblíbenou alternativou LSTM pro modelování sekvenčních dat, zejména v případech, kdy jsou omezené výpočetní zdroje nebo je dostačující jednodušší architektura. GRU také řeší problém mizejícího gradientu (hodnoty používané k aktualizaci vah sítě), kterým trpí klasické RNN. Pokud se gradient v průběhu času při zpětném šíření zmenšuje, může se stát příliš malým na to, aby ovlivnil učení, a neuronová síť se tak stane netrénovatlnou. Pokud se vrstva v neuronové síti nemůže učit, mohou RNN v podstatě „zapomenout“ dlouhé sekvence.

Stejně jako jiné architektury RNN zpracovává GRU sekvenční data po jednom prvku a aktualizuje svůj skrytý stav na základě aktuálního vstupu a předchozího skrytého stavu. Jak již bylo zmíněno GRU pak pomocí bran vypočítá průběžný aktivační vektor, který kombinuje informace ze vstupu a předchozího skrytého stavu. Tento průběžný vektor se pak použije k aktualizaci skrytého stavu pro další časový krok.

Brána aktualizace a resetovací brána jsou počítány z aktuálního vstupu a předchozího skrytého stavu pomocí rovnice:

$$r_t = sigmoid\big(W_r \cdot (h_{t-1}, x_t)\big) \tag{25}$$

$$z_t = sigmoid\big(W_z \cdot (h_{t-1}, x_t)\big) \tag{26}$$

kde $r$ je resetovací brána, $z$ je brána aktualizace, $x$ je aktuální vstup, $h_{t-1}$ je předchozí skrytý stav, $W_r$ a $W_z$ jsou váhové matice naučené během tréninku.

Průběžný aktivační vektor $h_t$ je počítán pomocí aktuálního vstupu a upravené verze předchozího skrytého stavu resetovaného resetovací bránou:

$$h_t' = tanh\big(W_h \cdot (r_t \cdot h_{t-1}, x_t)\big) \tag{27}$$

kde $W_h$ je další váhová matice.

Nový skrytý stav je dále vypočítán pomocí průběžného aktivačního vektoru s předchozím skrytým stavem vážený aktualizační bránou:

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot h_t' \tag{28}$$

Výsledkem je kompaktní architektura, která je schopna selektivně aktualizovat svůj skrytý stav na základě vstupu a předchozího skrytého stavu, aniž by bylo nutné vytvářet samostatný stav paměťové buňky jako v LSTM.

Architektura GRU se tedy skládá z následujících komponentů:

1. Vstupní vrstva – přijímá sekvenční data, jako jsou posloupnost slov nebo časová řada hodnot, a posílá je do GRU.
2. Skrytá vrstva – dochází zde k rekurentním výpočtům, kdy se aktualizuje v každém časovém kroku skrytý stav na základě aktuálního vstupu a předchozího skrytého stavu. Skrytý stav je vektor čísel, který představuje „paměť“ sítě předchozích vstupů.
3. Resetovací brána – určuje, kolik stavu se má zapomenout z předchozího skrytého stavu. Na vstupu přijímá předchozí skrytý stav a aktuální vstup a vytváří vektor čísel mezi 0 a 1, který určuje, do jaké míry se přechozí skrytý stav v aktuálním časovém kroku resetuje.
4. Brána aktualizace – určuje, jak velká část průběžného aktivačního vektoru má být začleněna do nového skrytého stavu. Jako vstup se bere předchozí skrytý stav a aktuální výstup a vytváří se vektor čísel mezi 0 a 1, který řídí míru začlenění průběžného aktivačního vektoru do nového skrytého stavu.

5. Průběžný aktivační vektor – jedná se o upravenou verzi předchozího skrytého stavu, který je resetován resetovací branou a zkombinován s aktuálním výstupem. Vypočítá se pomocí aktivační funkce tanh, jejíž výstup je mezi -1 a 1.
6. Výstupní vrstva – přijímá jako vstup konečný skrytý stav a vytváří výstup sítě, tím může být jedno číslo, posloupnost čísel nebo rozdělení pravděpodobnosti nad třídami, v závislosti na dané úloze.

Obecně lze tedy říct, že GRU sítě jsou mocným nástrojem pro modelování sekvenčních dat, zejména v případech omezených výpočetních zdrojů, nebo při požadované jednodušší architektuře. Jako každý model strojového učení má svoje výhody i nevýhody a vyžaduje tedy při volbě modelu jejich zvážení vzhledem ke konkrétní úloze.

### 5.2.3 LONG SHORT – TERM MEMORY (LSTM) NEURONOVÁ SÍŤ

Jedná se o speciální druh RNN, který se dokáže učit dlouhodobé závislosti. LSTM fungují v různých problémech a v současné době jsou hojně využívány. Stejně jako GRU i LSTM řeší problém mizejících gradientů díky své jedinečné struktuře bran.

V architektuře LSTM jsou 3 hlavní brány [22]. Zapomínací brána – rozhoduje o tom, která informace se zapomene ze stavu buňky. Vstupní brána – ta naopak zapisuje novou informaci do stavu buňky. A nakonec výstupní brána – ta určuje, jaký bude další skrytý stav. V LSTM se využívá funkce sigmoidní a funkce tanh. Funkce tanh je využívána primárně pro vstupní bránu pro škálování hodnot a normalizaci mezi -1 a 1, přičemž normalizace je klíčový krok pro stabilizaci sítě v čase. Dále se tanh využívá pro výstupní bránu opět pro normalizaci pro vytvoření konečného výstupu. Sigmoidní funkce se využívá pro binární rozhodování. Hodnota 0 znamená „zapomenout“ nebo „nepropustit“, zatímco hodnota 1 znamená „zapamatovat“ nebo „propustit“. Díky tomu lze efektivně ovlivňovat hodnotu stavu buňky.

Příklad užití LSTM může být následující: Prvním krokem LSTM je rozhodnout, jaké informace ze stavu buňky se vyhodnotí. Toto rozhodnutí provádí sigmoidní vrstva. Dalším krokem je rozhodnutí, jaké nové informace se uloží do stavu buňky. Rozhodnutí má dvě části, nejprve musí sigmoidní vrstva rozhodnout, jaké hodnoty budeme aktualizovat, a poté vrstva tanh vytvoří vektor nových kandidátních hodnot, které by mohly být přidány do stavu buňky. V dalším kroku se tyto dvě vrstvy spojí a vytvoří aktualizaci stavu. Nakonec se musí rozhodnout o tom, jaký výstup se bude vypisovat. Výstup vychází ze stavu buňky, ale bude filtrován. Nejprve se tedy spustí sigmoidní vrstva, která rozhodne, které části stavu buňky se budou vypisovat, a poté se stav buňky prožene přes tanh funkci a vynásobí výstupem sigmoidní vrstvy tak, aby byly na výstupu pouze ty části, které zde mají být. Popis příkladu je princip funkce základní LSTM. Ve skutečnosti se však vlastně každá práce s LSTM trochu liší v drobných detailech.

LSTM nabízejí robustní způsob práce se sekvenčními daty, který překonává omezení klasických RNN. Jejich schopnost zapamatovat si dlouhodobé závislosti z nich dělá volbu pro mnoho aplikací v dnešním světě řízeném umělou inteligencí.

## 5.3 DOPŘEDNÁ NEURONOVÁ SÍŤ

Dopředná neuronová síť (FNN) je jedním z nejjednodušších typů umělých neuronových sítí. V této síti se informace pohybují pouze jedním směrem – dopředu, ze vstupních neuronů přes neurony skryté až do neuronů výstupních. V síti se neobjevují žádné smyčky ani cykly [23]. Dopředné neuronové sítě byly prvním typem vynalezených umělých neuronových sítí a jsou jednodušším než jejich protějšky – rekurentní a konvoluční neuronové sítě. Tento typ neuronových sítí se používá především v případech, kdy data, která se mají naučit, nejsou ani sekvenční, ani časově závislá.

Nejjednodušším typem dopředné sítě je již zmíněný Perceptron, dopředná neuronová síť bez skrytých neuronů. Perceptron má pouze vstupní a výstupní vrstvu. Výstupní neurony se počítají přímo ze součtu součinu jejich vah s odpovídajícími vstupními neurony s určitým zkreslením.

Ovšem Minsky a Papert [24] ukázali, že jednovrstvá dopředná neuronová síť nemůže řešit problémy, v nichž data nejsou lineárně oddělitelná, jako je například problém XOR. Přidání jedné nebo více skrytých vrstev do neuronové sítě však umožňuje tyto problémy řešit. Podle univerzální aproximační věty může FNN s jednou skrytou vrstvou reprezentovat libovolnou funkci [25]. Ačkoli v praxi je natrénování takového modelu velmi obtížné, a proto se obvykle přidává více skrytých vrstev pro řešení složitějších problémů.

Ve vícevrstvé neuronové síti máme vstupní vrstvu, výstupní vrstvu a jednu nebo více vrstev skrytých. Vstupní vrstva má tolik neuronů, kolik je rozměr vstupních dat. Počet neuronů ve výstupní vrstvě závisí na typu problému, který se neuronová síť snaží vyřešit. Čím více skrytých vrstev FNN obsahuje, tím složitější funkce může síť odhadovat. To je však spojeno s prodloužením doby trénování a zároveň tedy i zvýšení pravděpodobnosti přetrénování. K přetrénování dochází tehdy, zachytí-li model detaily trénovacích dat, dobře na trénovacích datech funguje, ale není schopen správně fungovat na datech testovacích, která nebyla užita při trénování. Existují ovšem regularizační techniky, které tomuto jevu dokáží zabránit.

Pro navržení systému tak, aby nám poskytl správný výstup pro zadaný vstup je nezbytné najít optimální váhy spojující jednotlivé vrstvy. To se provádí ve fázi trénování, která zahrnuje dva kroky. První – dopředné šíření informací a druhý – zpětné šíření informací. V první části se počítá výstup systému, který se následně porovná se správným výstupem a vrátí informaci o chybě odhadu. V části zpětného šíření se dle chyby odhadu mění váhy a snaží se chybu minimalizovat. Potom se opět spustí první krok s dopředným šířením a stále dokola, dokud není nalezena optimální sada vah pro daný problém.

### 5.3.1 MATEMATICKÁ INTERPRETACE DOPŘEDNÉHO ŠÍŘENÍ

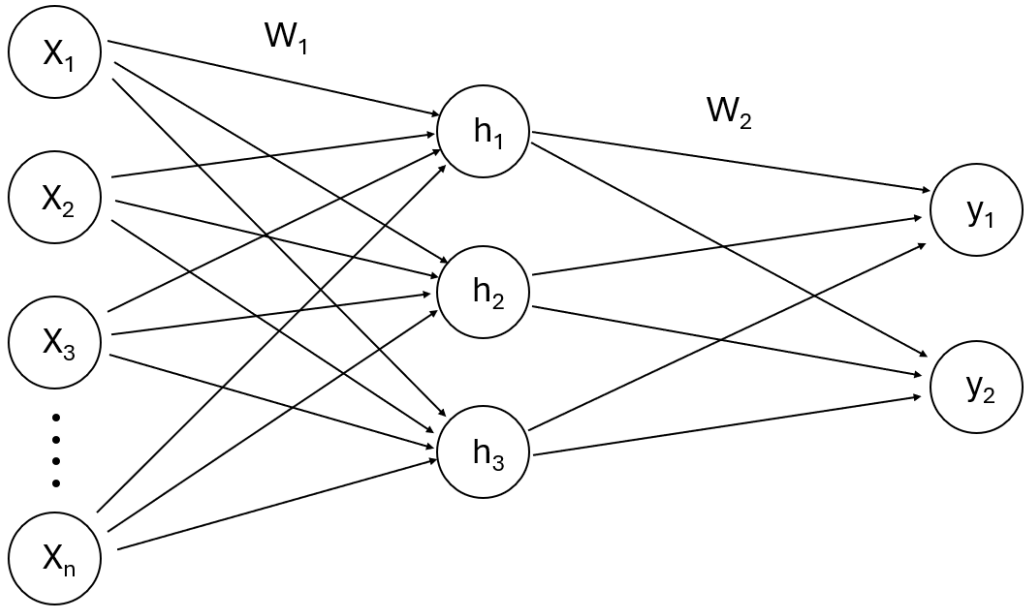
Pro zjednodušení problému bude uvažováno *n* vstupů, jedna skrytá vrstva se třemi neurony a dva výstupy viz *obr. 16*. V reálném problému může být *n* vstupů, *n* skrytých vrstev s mnoha neurony a *n* výstupy. Jelikož je uvažována pouze jedna skrytá vrstva, budou v každém cyklu dopředného šíření pouze dva kroky – hledání vektoru skryté vrstvy a hledání výstupního vektoru.

Vstupní i výstupní vrstva jsou reprezentovány více než jedním neuronem a zobrazují se tedy jako vektory. Kromě použití nelineárních aktivačních funkcí zahrnují všechny výpočty lineární kombinaci vstupů a vah, využívá se tedy násobení vektoru s maticí [26]. Stejně jako ve vstupní

a výstupní vrstvě je ve skryté vrstvě více neuronů a jedná se tedy o vektor. Každý vstupní neuron je připojen ke každému neuronu ve skryté vrstvě. Vektor ***h'*** skryté vrstvy se vypočítá vynásobením vstupního vektoru váhovou maticí ***W1*** následujícím způsobem:

$$[h_1' \quad h_2' \quad h_3'] = [x_1 \quad x_2 \quad x_3 \ldots x_n] \cdot \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ \vdots & \vdots & \vdots \\ W_{n1} & W_{n2} & W_{n3} \end{bmatrix} \tag{29}$$

Kde $h_1'$, $h_2'$ a $h_3'$ jsou složky vektoru skryté vrstvy, $x_1$ až $x_n$ jsou složky vektoru vstupů a $W_{11}$ až $W_{n3}$ jsou složky váhové matice.




*Obr. 16* Schéma dopředné neuronové sítě

Po nalezení vektoru ***h'*** je kvůli předejití explodování nebo přílišného zvětšení hodnoty potřeba aktivační funkce, kterou se dokončí výpočet hodnot skryté vrstvy. Aktivační funkcí $\Phi$ může být hyperbolický tangens, sigmoid nebo funkce ReLU.

Finální vektor hodnot skryté vrstvy ***h*** se vypočítá:

$$\boldsymbol{h} = \boldsymbol{h}'\phi \tag{30}$$

Jelikož $W_{ij}$ je váhová složka ve váhové matici, která spojuje neuron *i* ze vstupu s neuronem *j* ve skryté vrstvě, můžeme rovnici dále rozepsat:

$$\begin{aligned} \mathrm{h}_1 &= \phi(x_1 W_{11} + x_2 W_{21} + \cdots + x_n W_{n1}) \\ \mathrm{h}_2 &= \phi(x_1 W_{12} + x_2 W_{22} + \cdots + x_n W_{n2}) \\ \mathrm{h}_3 &= \phi(x_1 W_{13} + x_2 W_{23} + \cdots + x_n W_{n3}) \end{aligned} \tag{31}$$

Kde $h_1$, $h_2$ a $h_3$ jsou složky finálního vektoru skryté vrstvy.

Vzorec jde dále ještě zobecnit do výrazu:

$$\mathrm{h_m} = \phi\left(\sum_{i}^{n} x_i W_{im}\right) \tag{32}$$

Kde $\Phi$ je aktivační funkce, $x_i$ je i – tá složka vektoru vstupů a $W_{im}$ je i – tá složka spojující matice.

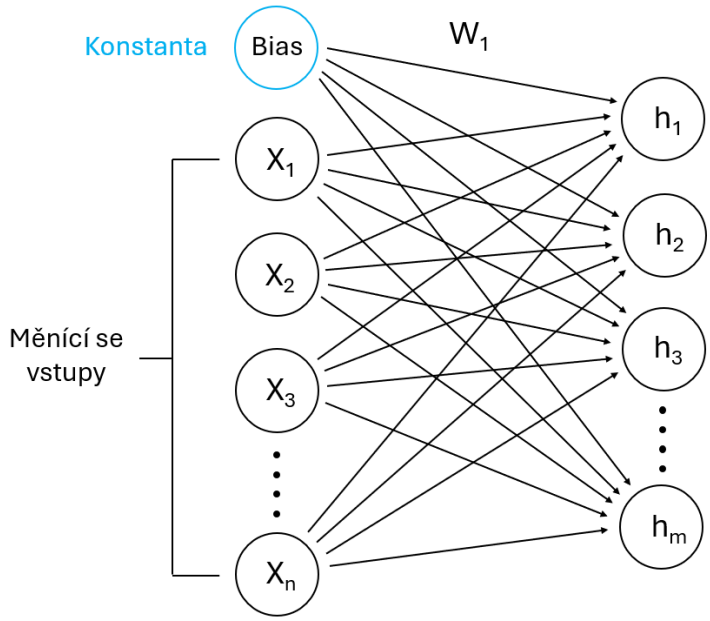
Postup výpočtu výstupního vektoru je podobný postupu výpočtu skryté vrstvy. Opět se využívá násobení vektoru maticí, za kterým může následovat aktivační funkce. Vektor je v tomto případě vypočtený vektor skryté vrstvy a matice je ta, která spojuje skrytou vrstvu s výstupní vrstvou. Tento postup by šel v podstatě zobecnit na jakoukoliv další vrstvu, kdy se vynásobí vektor s maticí, kde vektor představuje vstupy do nové vrstvy a matice spojuje tyto vstupy s další vrstvou.

Ve zjednodušeném problému viz *obr. 17.* je tedy vektor $\boldsymbol{h}$ a spojující matice $\boldsymbol{W_2}$.

$$[y_1 \quad y_2] = [h_1 \quad h_2 \quad h_3] \cdot \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \tag{33}$$

Kde $y_1$ a $y_2$ jsou složky výstupního vektoru $\boldsymbol{y}$ a $w_{11}$ až $w_{32}$ jsou složky váhové matice $\boldsymbol{W_2}$.

Aby bylo dosaženo dobré aproximace výstupu $\boldsymbol{y}$, je potřeba více úrovní skrytých vrstev. Může jich být tisíce. V podstatě se na tyto vrstvy dá dívat jako na stavební celky, které se dají skládat za sebe.




*Obr. 17* Schéma neurnové sítě s využitím biasu

Při použití biasu se nemění žádný z uvedených výpočtů. Jednoduše se bias uvažuje jako konstanta spojená ke každému z neuronu skryté vrstvy pomocí vah viz obr. *17*. Hlavní rozdíl mezi biasem a jakýmikoliv jinými vstupy je ten, že bias zůstává stále stejný. Ovšem stejně jako i u všech ostatních vstupů se aktualizují váhy, které bias spojují s další vrstvou.

Jelikož je cílem najít nejlepší sadu vah, která poskytne požadované výstupy pro dané vstupy, je nutné minimalizovat chybu odhadu. Je tedy nutné zjistit chybu v dané iteraci trénování. K tomu se nejčastěji využívají chybové funkce jako např. již zmíněná střední kvadratická chyba (RMSE).

$$\mathrm{h_m} = \phi\left(\sum_{i}^{n} x_i W_{im}\right) \tag{34}$$

### 5.3.2 BIAS A JEHO VYUŽITÍ V NEURONOVÉ SÍTI

Bias lze zjednodušeně definovat jako konstantu, která se přičítá k součinu funkcí a vah. Je jedinečný pro každou vrstvu neuronové sítě, pomáhá modelům posunout aktivační funkci směrem ke kladné nebo záporné straně a využívá se pro zpřesnění výsledků [27]. Pokud tedy dojde při predikci k nějaké chybě, lze k výstupním hodnotám přičíst hodnotu biasu pro získání skutečných hodnot.

Možné využití biasu je zahrnout jej přímo do aktivační funkce místo do výstupu. Pokud je použitá aktivační funkce v neuronové síti lineární, neurony budou často neaktivní, pokud jejich vstup nabývá určitých hodnot. Budeme-li uvažovat např. aktivační funkci ReLU, která je reprezentována následovně:

$$\mathrm{ReLU(x)} = \mathrm{y} = \max\left(0, \mathrm{x}\right) \tag{35}$$

Pro všechny hodnoty *x* menší nebo rovné 0 bude mít neuronová síť problém mizejícího gradientu. K tomu dochází tehdy, blíží-li se vstup nule nebo je-li záporný a gradient se blíží nule. Síť tedy nemůže provádět zpětné šíření a nemůže se tak učit. Tento problém se netýká pouze ReLU, ale i jiných aktivačních funkcí, jako je např. sigmoid, tanh atd. Pokud bychom do aktivační funkce zahrnuli bias, umožnilo by to neuronové síti posunout aktivační funkci doleva a doprava pouhou změnou hodnot biasu. To by dále umožnilo inicializaci derivací chybové funkce pro nenulové hodnoty *x* mezi 1 a 0. Díky tomu jsou aktivovány neurony a zpětné šíření i v případě nulového vstupu.

Je zde i několik možností, jak se vyhnout nutnosti použití biasu při tvorbě neuronové sítě. Mezi ty hlavní patří např.: výběr správného modelu strojového učení dle odhadovaných veličin, správná vyváženost tříd tréninkových dat, dobré zpracování vstupních dat nebo vyvarování se ztráty dat a jejich smyslu v průběhu učení.

## 5.4 VOLBA TYPU NEURONOVÉ SÍTĚ

Tato práce se bude zabývat odhadem dynamických veličin. S jistotou lze tedy říci, že se bude pracovat s naměřenými číselnými daty. Jelikož je cílem odhad číselných veličin na základě naměřených dat, bude se jednat o regresní úlohu. Znamená to tedy, že klasifikace nebude předmětem navrhované neuronové sítě a je možné vyloučit z výběru konvoluční neuronovou síť, která je k tomuto problému předurčena.

Vzhledem k tomu, že se jedná o veličiny dynamické, tedy veličiny fyzikální podstaty závislé na čase, nabízí se využít sítě RNN, která při trénování zohledňuje informace předchozích kroků. Důležité je ale vzít v potaz i to, zda se jedná o jakýsi sekvenční vzor, či nikoliv. Jelikož jsou naše data měřená v čase, je důležité, aby právě čas byl jednou ze vstupních veličin neuronové sítě.

Jak již bylo zmíněno, naměřená data jsou veličiny fyzikální podstaty, které jsou v praxi závislé na čase. Je tedy nutné vyřešit otázku, zda je důležité, aby o sobě dva stavy, které nastaly při měření vzájemně věděly. Pro úvahu se mohou vzít v potaz např. vlastnosti pneumatiky – ty se během jízdy, tedy se změnou času a působením okolních vlivů mění. Na začátku jízdy jsou pneumatiky studené a chovají se jinak než po určité době jízdy, kdy jsou již pneumatiky zahřáté. Pokud by bylo požadavkem odhadnout teplotu pneumatik v dané chvíli, pro neuronovou síť by bylo důležité vědět, zda se v minulých krocích pneumatika ochlazovala nebo naopak ohřívala. Touto úvahou je tedy možné říct, že pro řešení neuronové sítě pro predikci fyzikálních veličin je důležité vzít v potaz i minulé kroky.

Melzi a spol. [28] vytvořili tři neuronové sítě s danou architekturou, ale různými vstupy. V základu používají stejné vstupy jako Chindamo a spol. [29] bez minulých kroků, v dalším nastavení však rozšiřují vstupy právě o další minulé časové kroky vstupních veličin, což vede k přesnějším výsledkům. Spielberg a spol. [30] představili neuronovou síť pro nahrazení fyzikálního modelu vozidla. Model byl využit ke stanovení dopředného řízení u zkušebního autonomního vozidla. Vstupy byly podélná a boční rychlost, úhel odklonu, úhel natočení volantu a podélná síla na přední nápravě vozidla. Vedle aktuálních hodnot jsou ale použity jako hodnoty vstupní i informace ze tří minulých časových kroků.

S ohledem na vytvořené neuronové sítě lze konstatovat, že pro odhad dynamických veličin vozidel lze využít jak dopřednou neuronovou síť, tak síť RNN. Ovšem dle [28] a [30] byly při použití sítě RNN výsledky přesnější než při použití sítě dopředné. Jedná se tedy o jeden z důvodů, proč použít RNN síť.

# 6 ZPRACOVÁNÍ NAMĚŘENÝCH DAT

Zpracování dat je klíčovým krokem v návrhu neuronové sítě. Pro trénování sítě není ve většině případů možné použít přímo surová naměřená data. V reálném světě jsou totiž naměřená data značně znečištěná a poškozená šumem, neúplnými informacemi nebo třeba chybějícími hodnotami. Data mohou být agregována z různých zdrojů nebo mít různé formáty. Ve strojovém učení platí obecné pravidlo, že čím větší množství dat máme k dispozici, tím lepší modely můžeme trénovat. Způsob zpracování dat závisí na typu dat a architektuře neuronové sítě.

Zpracování (jindy také předzpracování) dat zahrnuje kroky, které musí být provedeny, aby byla data transformována nebo zakódována tak, aby je mohl stroj snadno analyzovat. Cílem zpracování dat je tak ulehčit trénování sítě a zlepšit její přesnost a preciznost díky snadné interpretaci vlastností dat.

## 6.1 ZÁKLADNÍ KROKY ZPRACOVÁNÍ DAT

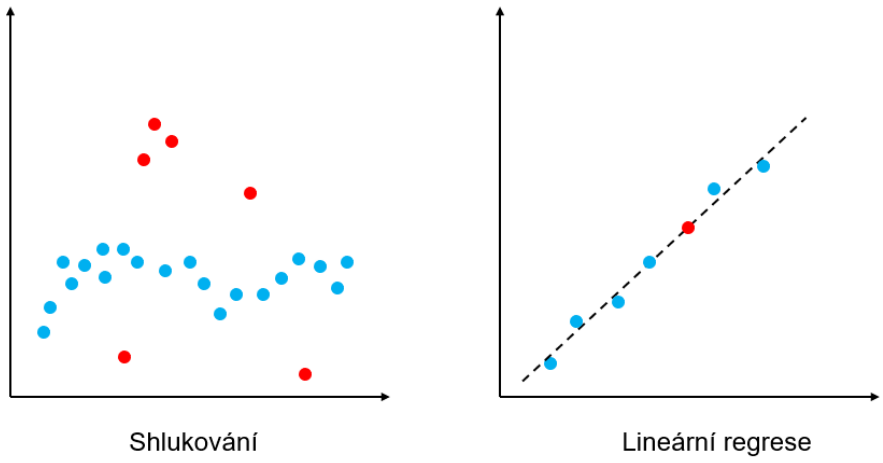
Většina reálných datových souborů pro strojové učení je velmi náchylná k chybějícím, zašuměným nebo nekonzistentním datům. Duplicitní nebo chybějící hodnoty mohou poskytnou nesprávný pohled na celkovou statistiku dat. Odlehlé a nekonzistentní datové body mají tendenci narušovat celkové učení modelu, což vede k chybným předpovědím. Kvalitní predikce musí vycházet z kvalitních dat. Jednotlivé nezávislé proměnné, které fungují jako vstupy do modelu neuronové sítě se často označují jako rysy. Lze si je představit jako vlastnosti nebo atributy, které popisují data a pomáhají modelům s predikcí. Například rysy ve strukturovaném souboru dat, jako je formát csv viz *obr. 18*, kde každý sloupec označuje měřitelnou část dat, kterou lze použít k analýze. Existují 4 základní kroky při zpracování dat [31]:

- čištění dat,
- integrace dat,
- transformace dat,
- redukce dat.

```
mpg, pocet_valcu, objem_ccm, pocet_koni, vaha_lbs, zrychleni_na_60, rok, vyroba
14,8,350,165,4209,12,1972, US.
31.9,4,89,71,1925,14,1980, Evropa.
17,8,302,140,3449,11,1971, US.
15,8,400,150,3761,10,1971, US.
30.5,4,98,63,2051,17,1978, US.
23,8,350,125,3900,17,1980, US.
13,8,351,158,4363,13,1974, US.
14,8,440,215,4312,9,1971, US.
25.4,5,183,77,3530,20,1980, Evropa.
37.7,4,89,62,2050,17,1982, Japonsko.
34,4,108,70,2245,17,1983, Japonsko.
34.3,4,97,78,2188,16,1981, Evropa.
16,8,302,140,4141,14,1975, US.
11,8,350,180,3664,11,1974, US.
19.1,6,225,90, ,19,1981, US.
16.9,8,350,155,4360,15,1980, US.
31.8,4,85,65,2020,19,1980, Japonsko.
```

*Obr. 18* Datový set ve formátu csv

Čištění dat je snaha o odstranění problémů odlehlých hodnot, šumu či chybějících hodnot. Při chybějících datech se obvykle hodnoty doplňují. To může být provedeno ručně, pomocí numerických metod nebo regresní funkcí viz *obr. 19,* kde modré body jsou naměřené hodnoty a červený bod je doplněná hodnota pomocí lineární regrese. Pokud existuje pouze jeden nezávislý atribut, používá se lineární regresní rovnice, jinak se využívají rovnice polynomiální. Je-li soubor dat obrovský, je možné chybějící data ignorovat. Je ovšem předem nutné zvážit, jak moc velký vliv to bude mít na trénování sítě. Odstranění šumu zahrnuje odstranění náhodné chyby nebo rozptylu v měřené proměnné. To lze provést např. pomocí techniky „binning“. Jedná se o techniku, která pracuje se setříděnými hodnotami dat. Data se rozdělí do stejně velkých „košů“ a s každým „košem“ se pak pracuje samostatně a vyhlazuje se případný šum, který se v datech vyskytuje. Všechna data v daném segmentu mohou být také nahrazena jejich průměrnou hodnotou, mediánem nebo hraničními hodnotami. Další technikou je shlukování pro odstranění šumu. Pomocí této techniky se postupně vytvářejí skupiny dat, které mají podobné hodnoty. Hodnoty, které neleží v žádném shluku se považují za zašuměné a odstraní se. Pro odstranění odlehlých hodnot se využívá stejné techniky shlukování viz *obr. 19,* kdy body, které leží mimo shluk (v tomto případě červené) se považují za odlehlé a opět se odstraní.




*Obr. 19* Metody shlukování (vlevo) a doplnění hodnot pomocí lineární regrese (vpravo); Metoda shlukování – červené body ležící mimo shluk jsou odstraněny; Metoda lineární regrese – červený bod je pomocí regrese doplněn

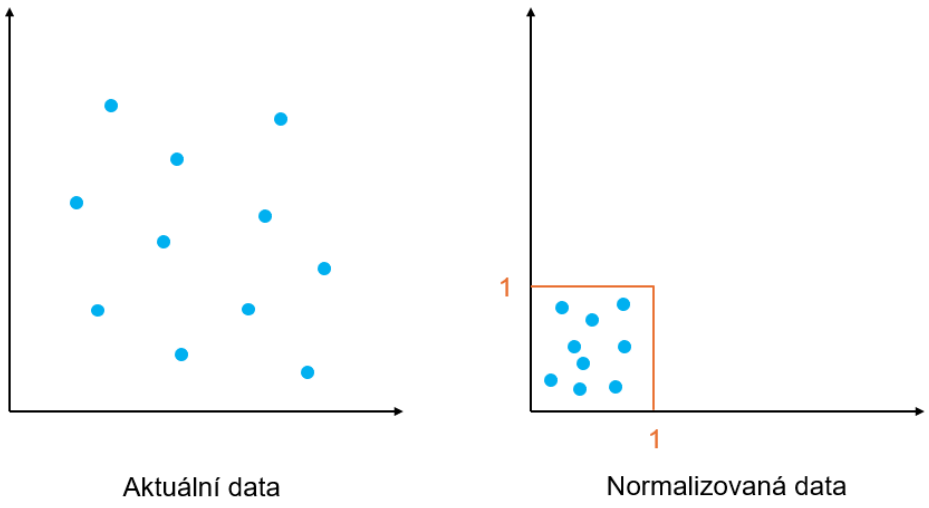
Integrace dat je krok, ve kterém jde o sloučení dat z více různých zdrojů do jednoho datového úložiště, jako datový sklad nebo databáze. Integrace je potřebná zejména tehdy, snažíme-li se vyřešit reálný problém, jako je např. detekce zlomeniny na rentgenovém snímku. Pro přesnější výsledky je dobré integrovat snímky z více lékařských institucí a vytvořit tak jednu velkou databázi. Při takové integraci se však mohou objevit další problémy. Data z různých institucí mohou mít různé formáty, které mohou při integraci způsobovat potíže, nebo mohou mít různé či nadbytečné atributy, které je potřebné odstranit.

Po vyčištění dat je třeba transformovat data do alternativních forem změnou jejich hodnoty, struktury nebo formátu pomocí různých strategií transformace dat. Jedna z používaných technik je generalizace, pomocí které se rozšiřuje obraz trendů nebo vzorců. Jedná se vlastně o zařazení dat do nadřazených skupin. Příklad generalizace je v *tab. 1*.

*Tab. 1* Generalizace dat

| Původní data | Generalizace |
|---|---|
| 16 | 10-19(2) |
| 19 | 20-29(3) |
| 21 | 30-40(6) |
| 24 | 40-50(5) |
| 25 | |
| 30 | |
| 30 | |
| 35 | |
| 38 | |
| 39 | |
| 40 | |
| 41 | |
| 44 | |
| 47 | |
| 47 | |
| 48 | |

Mezi nejdůležitější techniky transformace patří normalizace. Jedná se o hojně používanou techniku, která zmenší nebo zvětší číselné hodnoty tak, aby se vešly do určitého rozsahu viz *obr. 20*. Při tomto přístupu se omezuje atribut dat na určitý „kontejner“, abychom vytvořili korelaci mezi různými datovými body. Normalizaci lze provést vícero způsoby.




*Obr. 20* Normalizace dat do rozsahu hodnot 0 až 1

Další technikou je výběr atributů, kdy se z existujících atributů vytvářejí nové vlastnosti dat, které pomáhají zpracovat data a najít v nich vztahy či vzorce. Například „datum výroby automobilu“ může být přetransformováno do „jedná se o veterán“, což by mělo přímý dopad na odhad ceny vozidla. Poslední příklad metody transformace je agregace. Jedná se o metodu ukládání dat v souhrnném formátu, kdy například všechna data tržeb by se zobrazovala ve formátu za měsíc a za rok.

Dalším možným krokem zpracování je redukce dat. Může se stát, že velikost datového souboru v datovém skladu je příliš veliká pro algoritmy analýzy a zpracování dat. Potom je jedním z možných řešení redukce dat, kdy cílem tohoto kroku je sice zmenšit objem dat, ale zároveň zachovat stejně kvalitní analytické výsledky, které data poskytují. Redukce může být provedena např. metodou agregací datových kostek, kdy se shromážděná data vyjádří v souhrnné podobě.

Metoda redukce dimenzionality se používá k extrakci funkcí. Dimenzionalita datové sady se týká atributů nebo jednotlivých rysů dat. Cílem metody je snížit počet nadbytečných rysů uvažovaných v algoritmech strojového učení. Pokud lze po rekonstrukci již z komprimovaných dat získat data původní, označuje se komprese jako bezztrátová, v opačném případě se jedná o kompresi ztrátovou.

Diskretizace je metoda sloužící k rozdělení atributů spojitého charakteru na data s intervaly. Dělá se to proto, že spojité znaky mají většinou menší pravděpodobnost korelace s cílovou proměnnou a může být obtížnější výsledky interpretovat. Po diskretizaci lze interpretovat skupiny odpovídající přímo cílové proměnné. Např. čísla lze diskretizovat do jednotlivých skupin od 0 do 20, od 20 do 60 a nad 60.

Pomocí numerizační redukce lze data reprezentovat jako model nebo rovnici a ušetřit tak místo v paměti. Výběr podmnožiny atributů je posledním z uvedených příkladů metod. Účelem výběru atributů je být konkrétní a zaměřit se pouze na důležité atributy, které mají přidanou hodnotu k trénování modelu. V opačném případě by to mohlo vést ke vzniku dat s vysokou dimenzí, která se pak obtížně trénují kvůli problémům s přetrénováním.

Na konec zpracování dat je možno posoudit kvalitu dat. To zahrnuje statistické postupy, které je potřeba dodržet, aby se zjistilo, že jsou data v pořádku a nemají žádné problémy. Data musí být použitelná a dávat správné výsledky. Mezi hlavní kritéria kvality dat patří:

- úplnost bez chybějících hodnot atributů,
- přesnost a spolehlivost z hlediska informací,
- konzistence ve všech funkcích,
- zachování hodnoty dat.

Pokud se vyskytne v hodnocených datech nějaký problém, je nutné je vyprofilovat a identifikovat problémy kvality pro případnou opravu a čištění.

## 6.2 ZPRACOVÁNÍ DAT PRO NAVRŽENÍ NEURONOVÉ SÍTĚ

Tato práce se zabývá navržením neuronové sítě pro odhad dynamických veličin automobilu. Konkrétně bude snaha o odhad hodnot veličin směrové úchylky $\alpha$ popsané v kapitole 3.1.1 a podélných sil na kolech $Fx$ vozidla během celé jízdy. Odhad bude proveden na základě

naměřených dat jiných dynamických veličin vozidla. Je proto nutné se nejdříve podívat na změřená data a zpracovat je do podoby použitelné pro trénování sítě.

Data byla naměřena pomocí inerciální jednotky OXTS RT3002 na vozidle Volkswagen Transporter a byla měřena během jízdy, která trvala přibližně 45 minut. Za tuto dobu bylo naměřeno celkem 47 různých proměnných se vzorkovací frekvencí 0,01 sekundy. Znamená to tedy, že ke každé proměnné se naměřilo 272389 hodnot. Z hlediska počtu hodnot se nejedná o nijak veliký datový soubor.

Výhodou ovšem je, že se celý datový soubor dá z jednotky stáhnout přímo jako struktura dat do Matlabu, a je tedy možné data rovnou zpracovávat. Na *obr. 21* je stažená datová struktura.


*Obr. 21* Struktura naměřených dat vybraných veličin

Některé z naměřených dynamických veličin na *obr. 21* jsou:

- rychlost vozidla ve všech osách kartézské soustavy souřadnic,
- zrychlení ve všech osách kartézské soustavy souřadnic,
- úhel sklonu vozidla,
- úhel náklonu vozila,
- a další.

Pro trénování neuronové sítě samozřejmě není nutné využít všech naměřených veličin. O tom, jaké veličiny bude vhodné použít, je třeba rozhodnout při zpracování dat. Je dobré držet se již zmíněného obecného pravidla, že čím více je vstupních dat, tím lépe, ale na druhou stranu data musí být kvalitní a mít nějakou přidanou hodnotu, která pomůže zpřesnit výsledky navrhované sítě. Díky tomu, že jsou všechna naměřená data se stejnou vzorkovací frekvencí, není v tomto směru potřeba data upravovat. V opačném případě by bylo nutné data převzorkovat na stejnou vzorkovací frekvenci, ať již interpolací, extrapolací či jinou metodou.

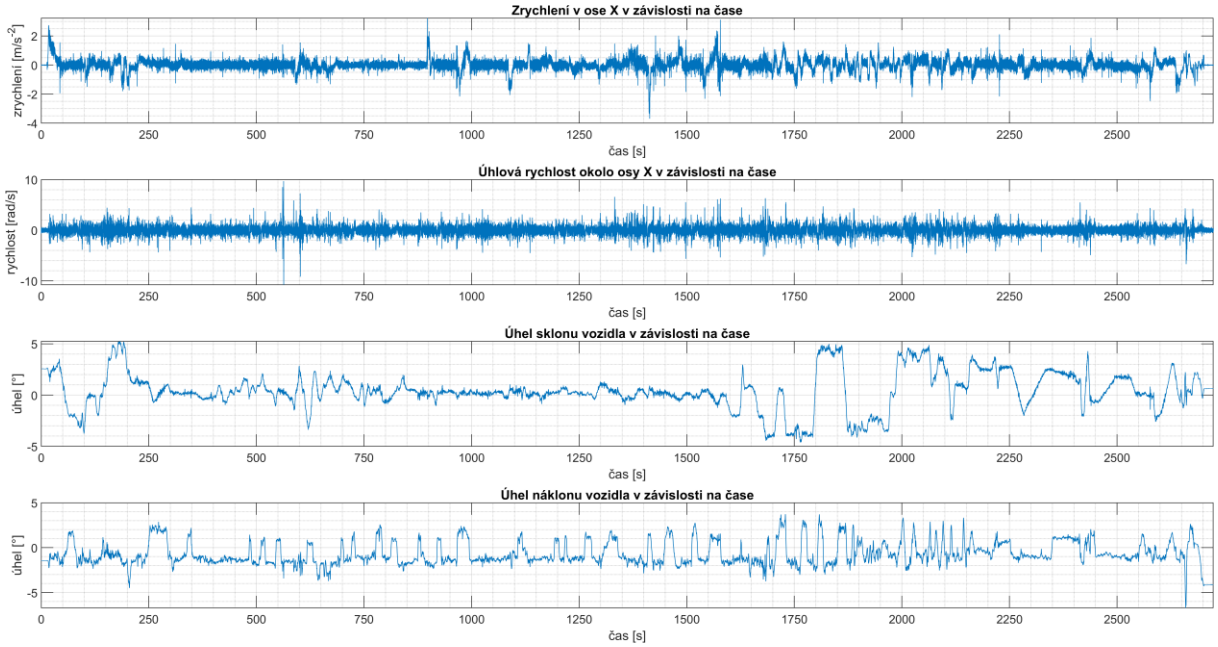
Co by se ovšem datům dalo vytknout je to, že byla měřena pouze na jednom typu vozidla. Tudíž i když se jedná o datový soubor, který může zahrnovat dostatečný počet různých stavů vozidla pro odhad veličin, je pravděpodobné, že síť nebude mít stejně dobré výsledky pro jiné typy vozidel.

Ze všech naměřených dat bylo pro začátek práce a jednodušší sestavení neuronové sítě vybráno 10 vstupních proměnných. Jsou to:

- zrychlení ve všech osách kartézského souřadného systému,
- úhel sklonu vozidla,
- úhel náklonu vozidla,
- úhlová rychlost okolo všech os kartézského souřadného systému,
- rychlost ve směru osy *y*,
- rychlost ve směru osy *z*.

Pro zjednodušení návrhu sítě bude nejdříve pouze jedna výstupní veličina, a to rychlost vozidla v podélném směru, tedy v ose *x*. Po úspěšném vytvoření neuronové sítě pro jeden výstup bude přidána i druhá výstupní veličina – směrová úchylka.

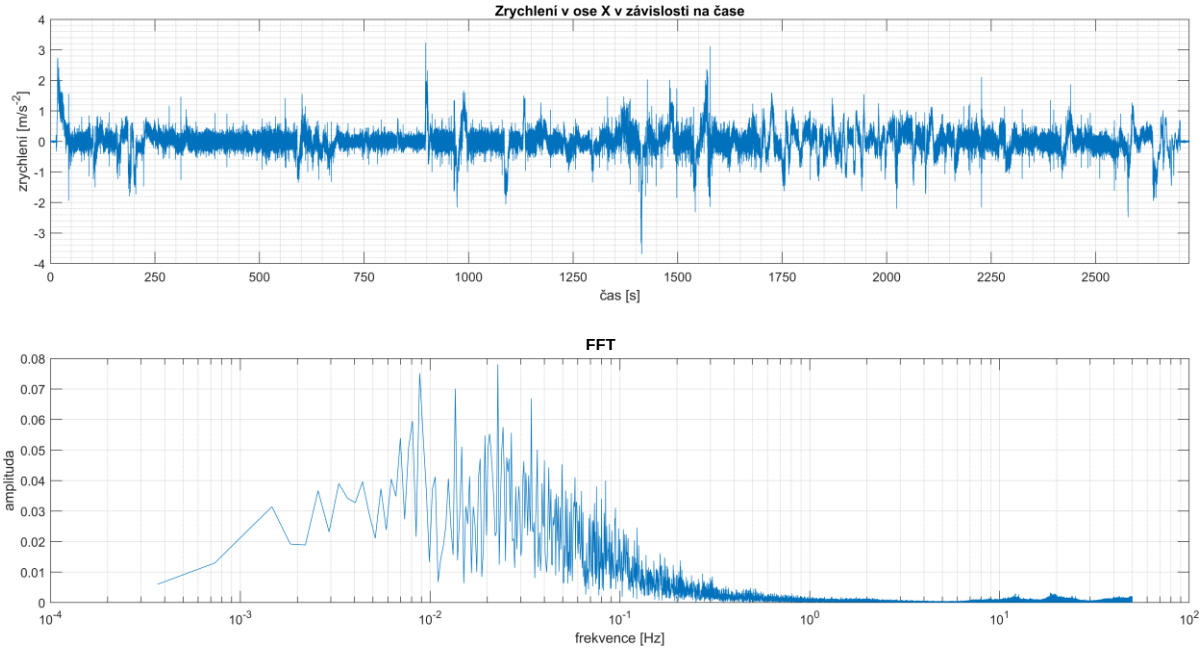
Na *obr. 22* je pro názornost zobrazen průběh vybraných naměřených veličin. Jedná se o neupravená data zrychlení v ose x, úhlové rychlosti okolo osy x, úhlu sklonu vozidla a úhlu náklonu vozidla. Všechny veličiny jsou vykresleny v závislosti na čase, tedy v průběhu 45minutové jízdy vozidla. Jednotlivé veličiny mají různé rozsahy hodnot.




*Obr. 22* Průběh naměřených vybraných veličin

Měření přechodných stavů vozidla se rozlišuje na různá frekvenční pásma. Pro boční a podélnou dynamiku je třeba uvažovat frekvenční rozsah od 0 do 10 Hz [32]. Při zkoumání vertikální dynamiky je frekvenční rozsah přibližně od 0 do 25 Hz. Kmitání s ještě vyššími frekvencemi pak způsobují především akustické problémy, které ale v tomto případě nejsou zajímavé, a proto nemusejí být brány v úvahu. Stejně tak pokud se pracuje s modelem

pneumatiky, který je schopen simulovat dynamické síly a momenty v pneumatikách, uvažuje se o veličinách v rozsahu frekvencí mezi 0 až 10 Hz [32][33]. Vzhledem k těmto poznatkům a k tomu, že vstupní veličiny jsou dynamické veličiny převážně z podélné a boční dynamiky, budou naměřená data filtrována právě do těchto frekvencí, tedy od 0 do 10 Hz.



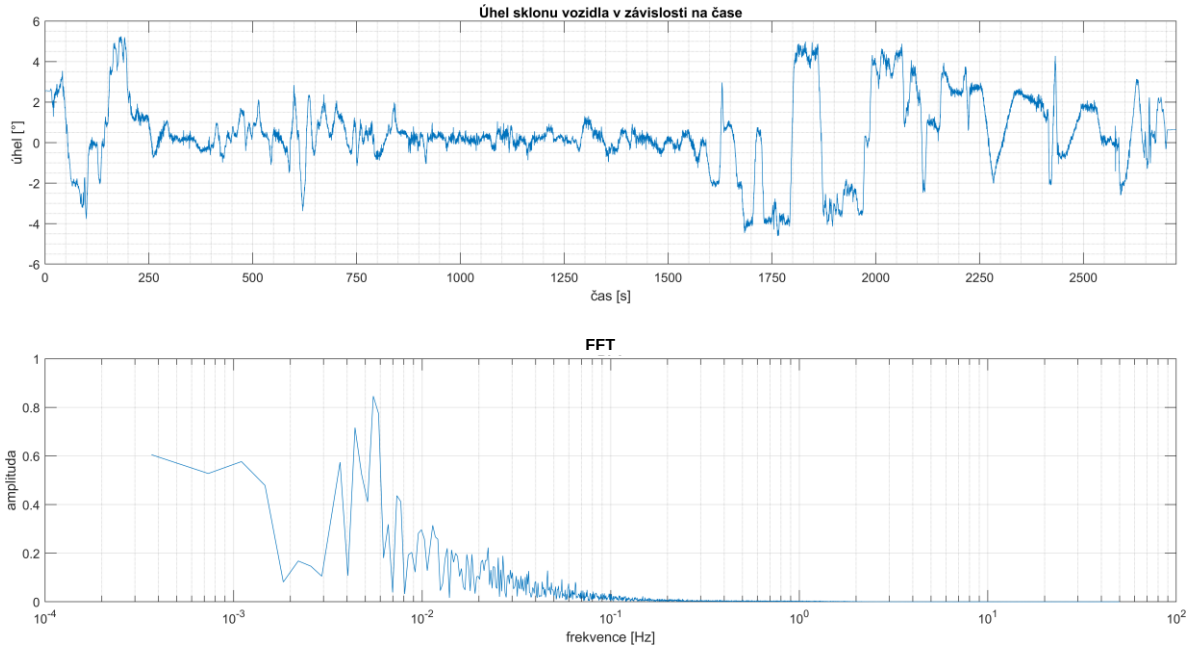

*Obr. 23* Převedení zrychlení v ose x z časové domény do frekvenční domény

Jak již bylo zmíněno, jelikož se jedná o veličiny naměřené v reálném provozu, data obsahují šum, který by mohl ovlivňovat výsledky trénované sítě. Z tohoto důvodu je potřeba data očistit od šumu. Pro získání představy o frekvenčním spektru naměřených veličin a případném vysokofrekvenčním šumu je využito rychlé Fourierovy transformace (FFT) k převodu časových řad dat na řady frekvenční.

Na *obr. 23* je na horním grafu zrychlení v ose x v časové doméně a na spodním grafu je již zrychlení převedeno do frekvenční domény. Pro zlepšení přehlednosti spodního grafu je osa x v logaritmickém měřítku. Rozsah frekvenčního spektra zrychlení je cca od $10^{-3}$ po $10^{2}$ Hz.

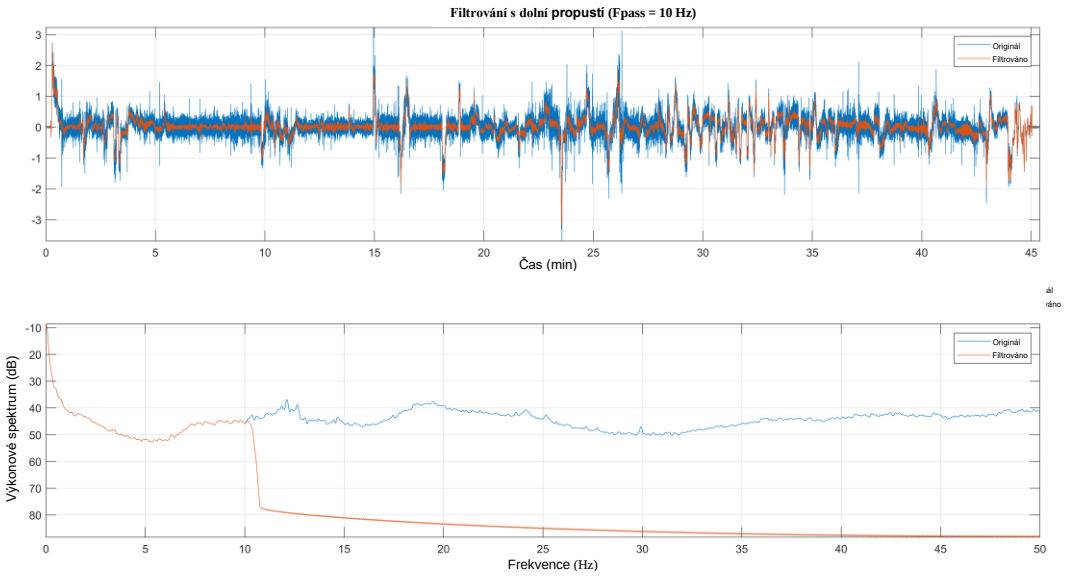
Pro názornost je zobrazena na *obr. 24* ještě další veličina – úhel sklonu vozidla. Opět je na horním grafu veličina v časové doméně a na spodním grafu je pomocí FFT převedena do frekvenční domény. Znovu je zde kvůli lepší přehlednosti logaritmické měřítko osy x. Rozsah frekvenčního spektra sklonu vozidla je zde podobný jako u rozsahu frekvenčního spektra zrychlení, tedy cca od $10^{-3}$ po $10^{2}$ Hz, i když s celkově větší amplitudou.

Po převedení signálu do frekvenční domény a ověření rozsahu frekvenčního spektra je nyní potřeba odstranit nežádoucí frekvence. Toho se docílí užitím správného datového filtru. Dle *obr. 23* a *obr. 24* je vidět, že naměřené frekvence jsou vyšší než 10 Hz, a proto bude pro filtraci využit filtr s dolní propustí (low-pass filter). Filtr s dolní propustí umožňuje sinusoidám s frekvencemi nižšími, než je specifikovaná mezní hodnota, projít filtrem bez ovlivnění, zatímco sinusoidy s frekvencemi vyššími, než je specifikovaná hodnota, jsou efektivně utlumeny a odstraněny z výstupního signálu.




*Obr. 24* Převedení úhlu sklonu vozidla z časové domény do frekvenční domény

Ačkoliv je funkce filtru s dolní propustí celkem známá, vliv na časový posun už tolik ne. Výstupní signál je obecně o nějakou hodnotu zpožděnou verzí signálu vstupního. Je to z toho důvodu, že filtr s dolní propustí s pevným řádem a mezní frekvencí zpožďuje sinusoidy různých frekvencí o různou hodnotu. Jedná se o nežádoucí, ale nevyhnutelný účinek filtrace. Ovšem časové zpoždění lze korigovat opětovným přefiltrováním již filtrovaných dat v opačném směru. Filtrování dat dvojitým průchodem zavádí stejné a opačné časové zpoždění, čímž se výstupní a vstupní signály časové řady znovu vyrovnají. Existují i určité případy, kdy může být při modelování události nutné zahrnout specifické časové zpoždění, ovšem v případě řešeném v této práci se jedná o nežádoucí účinek. Jelikož problém v této práci je řešen pomocí Matlabu, pro filtr s dolní propustí byla využita funkce „lowpass“ [34], která v sobě již zahrnuje i právě zmíněný dvojitý průchod a koriguje tak časový posun.



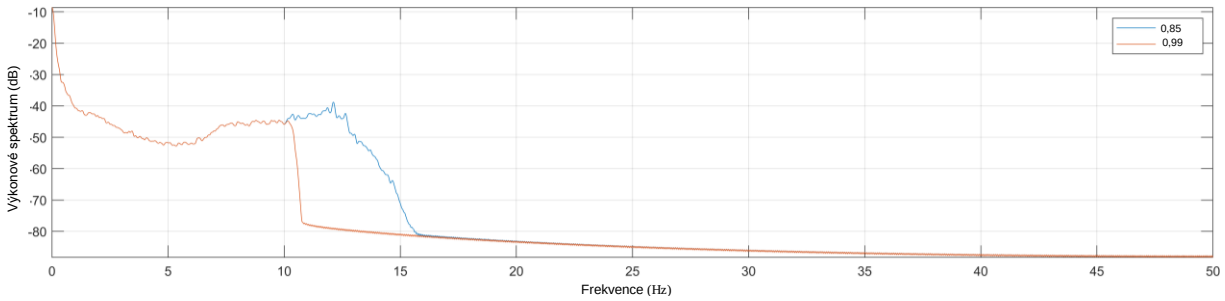

*Obr. 25* Grafy filtrování dat dolní propustí se specifikovanou mezní hodnotou 10 Hz

Na horním grafu na *obr. 25* je modře zobrazen průběh naměřených neupravených dat zrychlení v ose x v závislosti na čase a červeně zobrazen je průběh již filtrovaných dat dolní propustí s mezní hodnotou 10 Hz. Na spodním grafu je zobrazeno výkonové spektrum, kdy opět modře je průběh frekvenční charakteristiky naměřených neupravených dat a červeně průběh frekvenční charakteristiky filtru, přičemž je vidět, že je zde tzv. přechodová oblast filtru mezi cca 10 a 11 Hz. Přechodová oblast je rozsah frekvencí, který umožňuje přechod mezi propustným a zádržným pásmem filtru pro zpracování signálu. Zjednodušeně řečeno se jedná o oblast mezi místem, kde se filtr „ohýbá“ směrem dolů, a místem, kde se „dotýká dna“. Ideální filtr by neměl mít žádnou přechodovou oblast a měl by tedy v místě mezní hodnoty obsahovat ostré hrany. Cílem je tedy přiblížit se ideálnímu filtru. Zde se dostáváme k parametru strmost (stepness). Jedná se o skalár pohybující se v intervalu od 0,5 do 1. S rostoucím parametrem strmosti se odezva filtru blíží ideální odezvě, ovšem výsledná délka filtru a výsledné výpočetní náklady na filtrační operaci se zvyšují. V této práci byla hodnota parametru strmosti zvolena 0,99, což dle [34] odpovídá šířce přechodu o velikosti 1 % z $f_{Nyquist} - f_{pass}$.

Nyquistova frekvence je nejvyšší frekvenční složka signálu, kterou lze vzorkovat danou rychlostí bez zkreslení. Pro naměřená data s danou vzorkovací frekvencí se vypočítá jako:

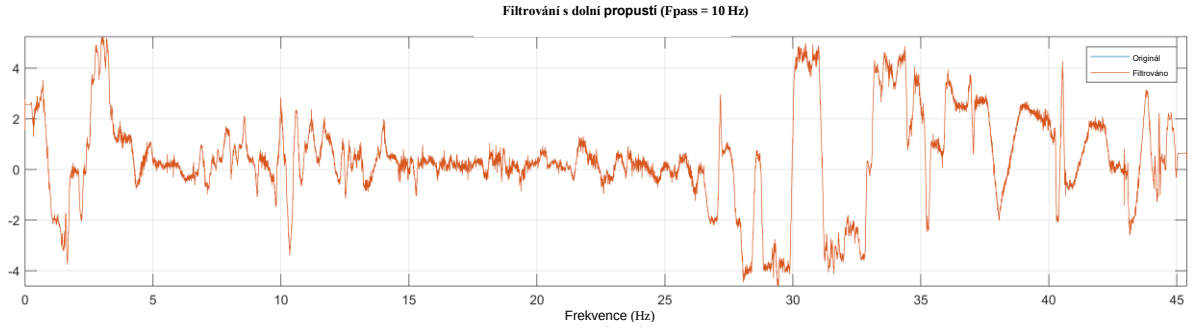
$$\mathrm{f_{Nyquist}} = \frac{f_s}{2} \tag{36}$$

Kde $f_{Nyquist}$ je Nyquistova frekvence a $f_s$ je vzorkovací frekvence měřeného signálu. Při dosazení čísel v tomto případě vychází $f_{Nyquist}$ = 50 Hz. Šířka přechodu pro strmost 0,99 tedy vychází jako 1 % z 40 Hz, což je 0,4 Hz. Na *obr. 26* je pro názornost zobrazeno, jaký je rozdíl v grafu, pokud je zvolena vyšší hodnota strmosti. Šířka přechodu je znatelně větší, což ovlivní i konečný výsledek upravených dat. Červeně je zobrazen průběh filtru se strmostí 0,99, která byla využita pro naměřená data v této práci a modře je zobrazen průběh filtru se strmostí 0,85.



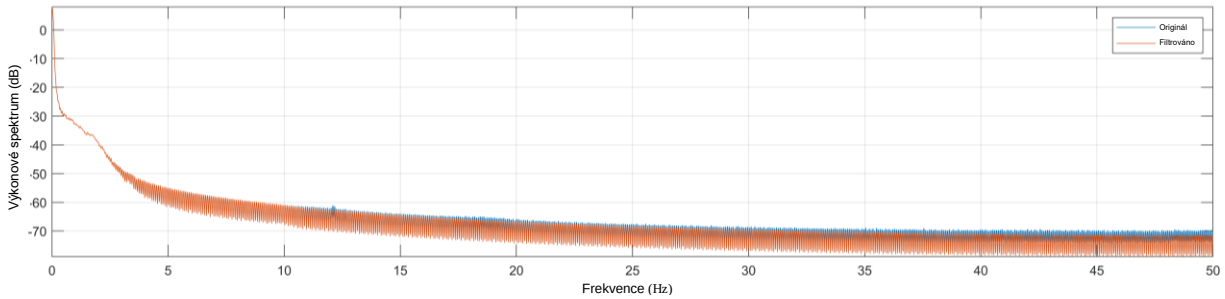

*Obr. 26* Rozdíl v průběhu frekvenční charakteristiky filtru s dolní propustí při volbě parametru strmosti

Pro lepší představu o průběhu upravených dat je na *obr. 27* zobrazeno použití filtru s dolní propustí pro další měřenou veličinu – úhel sklonu vozidla.




*Obr. 27* Průběh filtrovaných a nefiltrovaných dat naměřené veličiny úhlu sklonu vozidla

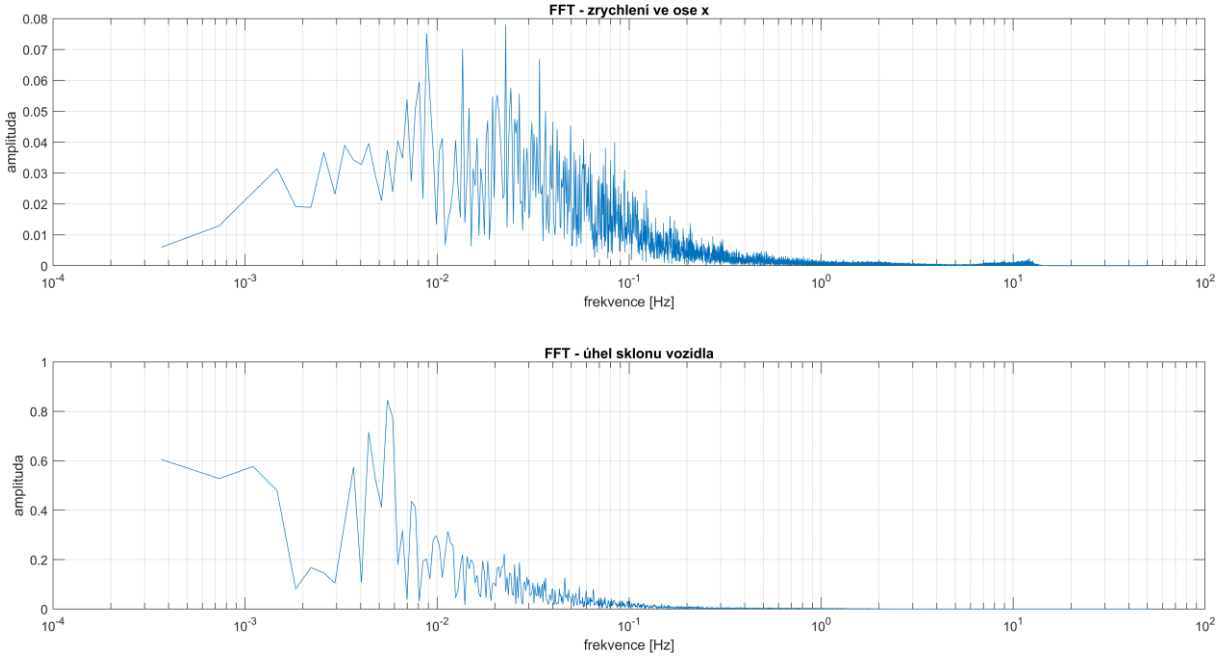
U této veličiny je možné pozorovat, že na ni filtrace měla jen malý, na první pohled přehlédnutelný, vliv. Je to z toho důvodu, že viz *obr. 24* je v měřených datech jen málo frekvencí úhlu sklonu vozidla nad 10 Hz. Modře (na *obr. 27* téměř neviditelně) je zobrazen průběh neupravených naměřených dat, zatímco červeně je průběh již filtrovaných dat. I u průběhu výkonového spektra filtru je možné pozorovat malý vliv změny viz *obr. 28*. Dále je možné si všimnout, že průběh frekvenční charakteristiky filtru se v závislosti na měřené proměnné liší, přestože nastavení filtru s dolní propustností zůstává stejné, tedy se specifikovanou hodnotou propustnosti 10 Hz a hodnotou parametru strmosti 0,99.




*Obr. 28* Výkonové spektrum filtru s dolní propustí se specifikovanou hodnotou 10 Hz

Pro ověření funkčnosti filtru s dolní propustí je využito inverzní Fourierovy transformace. Zatímco Fourierova transformace převádí funkci z časové domény do frekvenční domény, inverzní Fourierova transformace převádí funkci z frekvenční domény do časové. Pokud se tedy provede inverzní Fourierova transformace na data upravená pomocí filtru, měla by být v zásadě zpět na původních hodnotách souboru dat s tím rozdílem, že zde budou chybět hodnoty nad 10 Hz.

Na *obr. 29* jsou na obou grafech zobrazeny FFT již upravených veličin filtrem s dolní propustí. Horní graf zobrazuje zrychlení v ose x a spodní graf úhel sklonu vozidla. Oba grafy mají osu x v logaritmickém měřítku pro zlepšení přehlednosti. Z grafů je očividné, že frekvence od 10 Hz jsou potlačeny a filtr tedy zafungoval správně. Pokud bychom porovnali grafy s *obr. 23* a *obr. 24*, je zřejmé, že jsou průběhy frekvenčních spekter totožné s již zmíněným jediným rozdílem od 10 Hz. Lze si také povšimnout, že u veličiny zrychlení v ose x zde zůstalo pár frekvencí nad 10 Hz. Jedná se o frekvence, které jsou v onom přechodu filtru s dolní propustí.




*Obr. 29* FFT pro ověření funkčnosti filtru s dolní propustí

Vzhledem k tomu, že byla ověřena funkčnost filtru s dolní propustí a byl splněn účel využití, je filtr použit na všechny ostatní měřené veličiny. Následně byl ještě celý datový soubor překontrolován, zda u nějaké veličiny nechybí hodnoty, aby nebylo negativně ovlivňováno trénování sítě.

Posledním krokem zpracování dat je po úspěšné úpravě rozdělení datového souboru. Rozdělení datového souboru se dělá proto, aby se nepoužívala stejná sada dat pro trénování a hodnocení modelu. Pokud má být vytvořený spolehlivý model neuronové sítě, je nutné rozdělit data na trénovací, validační a testovací sadu. Pokud by nebyla rozdělena, mohla by být kvalita sítě falešně vyhodnocena a mohla by tak vytvářet dojem lepší přesnosti, než by ve skutečnosti byla.

Trénovací soubor dat je soubor dat, který se využívá k trénování sítě, aby se model mohl naučit skryté vzory nebo rysy v datech. V každé epoše jsou neuronové síti opakovaně předávána stejná trénovací data, aby model mohl pokračovat v učení a upravovat tak jednotlivé váhy. Počet epoch je stanoven během sestavování neuronové sítě. Tréninková sada je v podstatě nejdůležitější sada dat, jelikož to, jak bude síť přesná, závisí na datech, na kterých se síť trénuje. Je proto dobré, aby tréninková data měla diverzifikovanou sadu vstupů tak, aby byl model natrénován na co nejvíce různých scénářů a mohl tak předpovídat i ještě neviděný vzorek dat, který se může někdy v budoucnu objevit.

Validační soubor dat je další soubor, který se využívá při trénování sítě. Během tréninku nám říká, zda se trénink ubírá správným směrem. Neuronová síť se trénuje na trénovacích datech a současně po každé epoše provede vyhodnocení modelu na datech validačních. Hlavním důvodem validační datové sady je zabránit přeučení modelu. To znamená, že se trénovaná síť nesmí až moc přizpůsobit a naučit na trénovací data, aby nebyla funkční pouze na ně, ale aby dokázala naučené vzory zobecnit a provést tak přesnou predikci i na datech, které nikdy neviděla.

Testovací soubor dat je samostatný soubor, který se po natrénování sítě používá k jejímu testování. Poskytuje nezkreslenou konečnou metriku výkonnosti neuronové sítě z hlediska

přesnosti. Zjednodušeně řečeno se využívá pro zhodnocení toho, jak dobře a přesně sestavená a natrénovaná neuronová síť funguje.

Poměr rozdělení datové sady závisí na celkovém počtu vzorků přítomných v datové sadě. Pokud je třeba vyladit několik hyperparametrů, model neuronové sítě vyžaduje větší validační množinu, aby se optimalizoval výkon sítě. Pokud je důležité vyhnout se falešným předpovědím, je lepší provádět validaci v každé epoše. S nárůstem funkcí dat se zvyšuje počet hyperparametrů sítě, čímž se zesložiťuje model. V těchto případech se zachovává velký podíl trénovacích dat s daty validačními. Ovšem neexistuje žádné přesně stanovené a optimální procento rozdělení. Je třeba vytvořit takový poměr rozdělení, který vyhovuje požadavkům a potřebám vytvářeného modelu. Při hledání optimálního poměru se však mohou naskytnout problémy. Pokud je tréninkových dat málo, neuronová síť bude vykazovat vysokou odchylku při trénování. Dále může mít při malém množství validačních dat statistika vyhodnocení větší rozptyl. Je tedy třeba navrhnout optimální rozdělení jak na základě požadavků modelu, tak na velikosti použitelné datové sady.

Postupně byly zavedeny různé techniky pro rozdělení dat [35]. Mezi nejoblíbenější patří např. náhodné vzorkování, stratifikované vzorkování nebo křížová validace. Náhodné vzorkování je, technika, kdy je datová sada náhodně zamíchání a vzorky jsou pak náhodně zařazeny do trénovací, validační a testovací sady dle zvoleného procentuálního poměru. Metoda má však značnou nevýhodu, a to takovou, že funguje optimálně pouze pro třídově vyvážené datové sady. V případě nevyváženosti může pak způsobit takové rozdělení zkreslení. Tento problém však zmírňuje stratifikované vzorkování. Je zde totiž zachováno rozložení tříd v každé z trénovacích, validačních a testovacích sad. Křížová validace neboli K-násobná křížová validace je již robustnější technika pro rozdělení dat, kdy je model natrénován a vyhodnocen k-krát na různých vzorcích. Pokud by byla například požadována pětinásobná křížová validace, rozdělí se datová sada na 5 stejných nepřekrývajících se částí. Nyní se z částí 1–5 vytvoří pět kompletních datových souborů následujícím způsobem: Pro datovou sadu 1 se použije část 1 jako validační sada a z částí 2-5 se vytvoří sada trénovací. Pro datovou sadu 2 se použije část 2 jako validační sada a z částí 1, 3, 4 a 5 se vytvoří sada trénovací atd. Při použití křížové validace je neuronová síť vystavena různým rozložením dat a tím se zmírní případné zkreslení, ke kterému může dojít při výběru trénovacích dat. Ovšem i křížová validace může trpět stejným problémem jako náhodný výběr. Proto se může křížová validace kombinovat se stratifikovaným vzorkováním.

V této práci byl datový soubor rozdělen dle poměru 70:15:15, tedy 70 % dat z datového souboru jsou data trénovací, 15 % jsou data validační a zbylých 15 % jsou data testovací. Zároveň nejsou data nijak míchána, a to z toho důvodu, že se jedná o časovou sekvenci, kdy je důležité, aby i při učení byly jednotlivé hodnoty za sebou tak, v jakém pořadí byly v časové sekvenci naměřeny. Umožňuje to modelu učit se z minulých vstupních dat trénovací množiny.

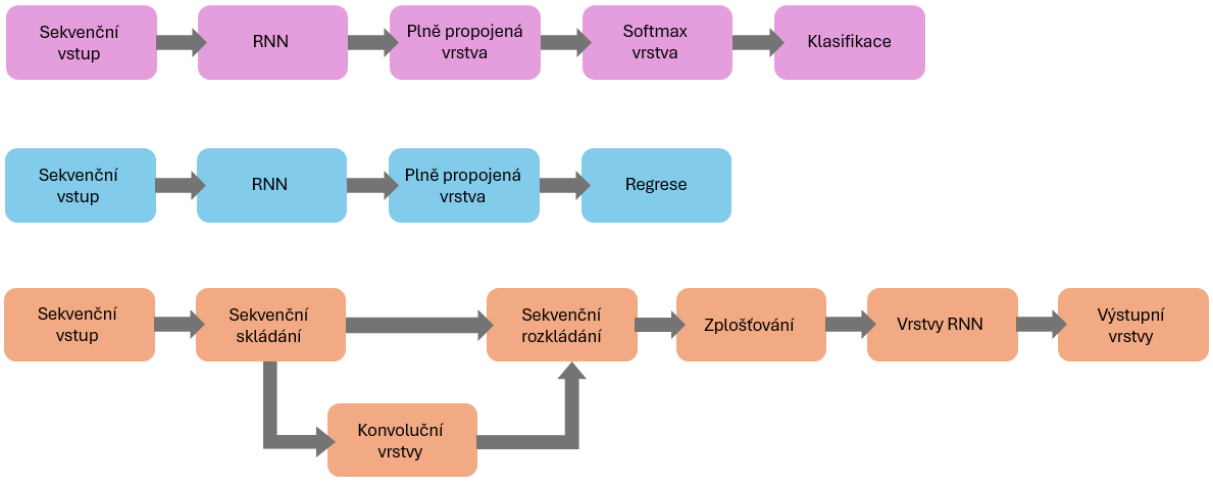
Pokud jsou trénovací data špatná, nelze očekávat, že bude model fungovat dobře. Vzhledem k tomu, že jsou algoritmy strojového učení na trénovací data citlivá, i malé chyby v trénovací sadě dat mohou vést k velkým chybám ve výkonu natrénované sítě. Přílišný důraz na metriky validačního a testovacího souboru také není dobrá. Metriky by měly být zvoleny tak, aby přinášely pozitivní vliv na celkovou trajektorii výkonosti sítě. Bohužel neexistuje přesné pravidlo nebo vzorec, který by zhodnotil kvalitu dat a jejich zpracování, a tak se zjišťuje až během trénování samotné neurnové sítě. Je tedy potřeba počítat s tím, že může být nutné v průběhu procesu trénování data upravit, převzorkovat či přerozdělit tak, aby s nimi mohla síť lépe pracovat.

# 7 SESTAVENÍ NEURONOVÉ SÍTĚ

Pro řešený problém v této práci, tedy pro odhad dynamických veličin pomocí neuronové sítě, byla vybrána rekurentní neuronová síť (RNN) viz kapitola 5.4. Pro navržení neuronové sítě je nutné se rozhodnout, jak sestavit jednotlivé vrstvy, aktivační funkce, které hyperparametry a případně jakou jejich hodnotu zvolit.

Rekurentní neuronové sítě mají 3 základní architektury, ze kterých je možno při sestavování vlastní sítě vycházet [36]. Volba architektury závisí na oblasti použití. Jedná se o architektury:

- klasifikace,
- regrese,
- video klasifikace.




*Obr. 30* Základní architektury RNN; Fialově zobrazená – klasifikační architektura, modře zobrazená – regresní architektura, oranžově zobrazená – architektura klasifikace videa

Jednotlivé architektury a jejich složení je zobrazeno na *obr. 30*. Klasifikační architektura se typicky používá pro klasifikační úlohy (například klasifikace e-mailu, zda se jedná o spam či nikoliv). Oproti ostatním základním architekturám je v tomto typu použita navíc softmax vrstva a klasifikační výstupní vrstva. Softmax vrstva se umisťuje na konec sítě, kde normalizuje výstupní hodnoty na rozmezí 0 a 1. Každá výstupní hodnota pak může být interpretována jako pravděpodobnost příslušnosti k daným třídám. Za softmax vrstvou následuje klasifikační vrstva, která produkuje výstupy vhodné pro klasifikaci. Architektura regresní se naopak používá pro regresní úlohy (například predikce číselných veličin). Na konci sítě se nachází regresní výstupní vrstva, která upravuje výstupy tak, aby byly vhodné pro regresi a daly se hodnotit metrikou určenou pro regresní úlohy. Architektura klasifikace videa je, jak již název napovídá, architektura sítě určená pro klasifikaci na základě obrazu či obsahu videa (např. detekce podezřelých aktivit). Tato architektura se od zbylých liší trochu více právě kvůli zpracování obrazu. Obsahuje sekvenční skládání, které se obvykle používá jako první krok. Během tohoto kroku jsou snímky videa seskupeny do pěvně definovaných časových bloků, čímž se vytvoří sekvence (série), které lze dále zpracovávat pomocí RNN. Sekvenční rozkládání

je dalším krokem, který umožňuje průchod dat po jednotlivých časových krocích a tím vytvoření zpětné vazby mezi těmito kroky. Dále se v architektuře nacházejí konvoluční vrstvy kvůli zpracování obrazu, zmíněné již v kapitole 5.1.1 a vrstva pro zplošťování, kde se vícedimenzionální data formátují do jednodimenzionálního vektoru, která byla opět popsána ve stejné kapitole. Na základě řešeného problému v této práci je pro řešení RNN z těchto základních architektur zvolena regresní architektura, ze které se bude vycházet dále.

### 7.1.1 POUŽITÉ HYPERPARAMETRY PRO NATRÉNOVÁNÍ RNN

Proces trénování zahrnuje výběr optimálních hyperparametrů, které algoritmy používají k dosažení co nejlepšího výsledku trénování. Hyperparametry jsou parametry, které jsou explicitně definovány uživatelem a slouží k řízení procesu učení. Předpona „hyper“ zde označuje, že se jedná o parametry nejvyšší úrovně. Jsou nastaveny před procesem trénování a jedná se tedy o externí hodnoty, které nelze během učení měnit. Mezi hyperparametry, které byly explicitně zadány pro natrénování sítě k predikci dat směrové úchylky patří:

- Optimizer (optimalizátor),
- Max Epochs (maximální počet epoch),
- Gradient Threshold (prahová hodnota gradientu),
- Initial Learn Rate (počáteční rychlost učení),
- Shuffle (náhodný výběr),
- Learn Rate Drop Factor (faktor poklesu rychlosti učení),
- Learn Rate Drop Period (perioda poklesu rychlosti učení),
- L2 Regularization (L2 regularizace),
- Validation Data (validační data),
- Validation Frequency (validační frekvence),
- Verbose (výpis),
- Plots (grafy),
- Execution Environment (prováděcí prostředí).

**Optimalizátor** neboli solver (řešič) je algoritmus, který minimalizuje chybu mezi očekávanými vstupy a výstupy neuronové sítě na základě poskytnutých trénovacích dat. Algoritmus provádí zpětnou propagaci chyby a upravuje váhy a biasy neuronů během trénování právě za účelem minimalizace chyby. V Matlabu se dají využít při sestavení vlastní sítě pro trénování 4 algoritmy:

- SGDM (Stochastický gradientní sestup s hybností)
- RMSProp (Střední kvadratické šíření)
- Adam (Adaptivní odhad momentu)
- L-BFGS (Omezená paměť – Broyden – Fletcher – Goldfarb – Shanno)

SGDM je varianta klasického stochastického gradientového optimalizátoru. Standardní algoritmus aktualizuje parametry sítě tak, aby minimalizoval ztrátovou funkci tím, že v každé

iteraci dělá malé kroky ve směru záporného gradientu ztráty [37]. Gradient ztrátové funkce se vyhodnocuje pomocí celé trénovací množiny. Oproti tomu SGDM vyhodnocuje gradient v každé iteraci a aktualizuje parametry pomocí podmnožiny trénovacích dat. Při každé iteraci se použije jiná podmnožina – tzv. minidávka. Projití celé trénovací množiny rozdělené na minidávky se nazývá epocha. Stochastický gradientní sestup je stochastický kvůli tomu, že aktualizace parametrů vypočtená pomocí minidávky je přibližným odhadem aktualizace parametrů, která byla výsledkem použití celé sady dat.

RMSProp je algoritmus, který se na rozdíl od SGDM, který používá pro všechny parametry stejnou rychlost učení, snaží přizpůsobit rychlost učení na základě parametrů a mění ji vzhledem ke ztrátové funkci. Algoritmus používá klouzavý průměr pro normalizaci aktualizací každého parametru zvlášť. Efektivně tak snižuje rychlost učení parametrů s velkými gradienty, a naopak zvyšuje rychlost učení parametrů s gradienty malými.

Optimalizátor Adam používá aktualizaci parametrů, která je podobná té v RMSProp, ale s přidaným členem hybnosti. Udržuje jak elementární klouzavý průměr gradientů parametrů, tak i jejich čtvercové hodnoty. Právě klouzavé průměry pak Adam používá k aktualizaci parametrů sítě. Pokud se stane, že jsou v průběhu iterací gradienty podobné, pak použití klouzavého průměru gradientu umožňuje, aby aktualizace parametrů nabývaly jednoho daného směru. Pokud jsou gradienty převážně zašuměné, pak se klouzavý průměr zmenšuje a tím se zmenšují i aktualizace parametrů. Úplný optimalizátor obsahuje také mechanismus pro korekci zkreslení, který se objevuje na začátku trénování.

L-BFGS je nejnovější algoritmus, který byl přidán poměrně nedávno ve verzi Matlabu 2023b [37]. Jedná se o kvazi – Newtonovu metodu a byl přidán za účelem použití pro malé sítě a datové sady, které lze řešit v jedné dávce.

Hyperparametr **maximální počet epoch** určuje již dle názvu maximální počet iterací procesu trénování. Jedna epocha představuje jedno projití tréninkových dat. Během každé epochy jsou váhy aktualizovány na základě chyby mezi predikcemi a skutečnými hodnotami. Specifikace maximálního počtu epoch má vliv na dostatečné natrénování a přetrénování. Pokud se síť příliš přizpůsobí tréninkovým datům a ztratí schopnost generalizace na nová data, je možné zmenšit efekt přetrénování snížením počtu trénovacích epoch. Na druhou stranu je nutné mít dostatečný počet epoch na to, aby se síť dokázala naučit a dosáhnout požadované přesnosti, je tedy potřeba najít správný balanc mezi oběma stavy.

Dalším zmíněným hyperparametrem je **prahová hodnota gradientu**. Tato hodnota určuje maximální hodnotu gradientu, kterou váhy sítě mohou mít. Pokud váhy přesáhnou stanovenou hodnotu, jsou ořezány na hodnotu maximální, tedy na hodnotu stanovenou. Hlavní funkcí hyperparametru je ovlivňovat gradienty vah během zpětné propagace chyby. Nastavením prahové hodnoty gradientu lze omezit velké gradienty a stabilizovat tak průběh tréninku. Dále slouží jako prevence k explozi gradientů a k omezení prudkých změn vah. Čím vyšší je nastavená hodnota hyperparametru, tím méně se váhy aktualizují a tím stabilnější je proces trénování a menší riziko exploze gradientů. Pokud bude hodnota příliš nízká, váhy se mohou aktualizovat tak málo, že se síť nedokáže natrénovat.

**Počáteční rychlost učení** je hyperparametr určený k nastavení hodnoty rychlosti učení, s jakou se váhy aktualizují během tréninku. „Počáteční“ rychlost učení je to proto, že se rychlost učení může v závislosti na průběhu optimalizace měnit. Jelikož ovlivňuje, jak rychle nebo pomalu se síť učí, efektivně tak ovlivňuje konvergenci procesu učení. Nastavení vhodné hodnoty je tedy klíčové pro úspěšný trénink sítě. Čím vyšší je hodnota hyperparametru, tím rychleji síť konverguje, ale tím méně stabilní při trénování je. Při vysoké rychlosti učení má také síť tendenci se přeučit. Naopak při příliš nízké hodnotě síť nemusí vůbec zkonvergovat. Ovšem velikost hyperparametru je relativní k řešenému problému. Při volbě vhodné hodnoty je tedy potřeba vzít v úvahu velikost datasetu, hloubku sítě a složitost problému. Obecně jsou menší datasety citlivější na vyšší rychlost učení. Složitější problémy nebo hluboké sítě vyžadují pomalejší rychlost učení, aby síť našla a naučila se i komplexnější spojitosti a vzorce v datech.

Hyperparametr **náhodný výběr** ovlivňuje způsob, jakým jsou tréninková data vybírána a používána v každé epoše. Hyperparametr nastavuje, zda jsou data promíchána nebo ne. Náhodný výběr může být nastaven na: „nikdy“ – pak se data nikdy nepromíchají; „jednou“ – data se náhodně promíchají před procesem trénování; „každou epochu“ – data se promíchají před každou epochou. Je důležité zmínit, že se promíchá jak trénovací, tak validační dataset. Promíchání dat pomáhá s generalizací dat a omezení vlivu pořadí. Na druhou stranu, pokud jsou data zamíchána ve specifických situacích, kdy je pořadí dat klíčové, jako například při práci s časovými řadami, kde pořadí samotných dat může obsahovat klíčové informace, může to způsobit potíže s dosažením dostatečné přesnosti neuronové sítě.

Pokud se nastaví plán učení sítě na „po intervalech“ přibydou další dva hyperparametry, které je možno nastavit. Jedná se o **faktor poklesu rychlosti učení** a **perioda poklesu rychlosti učení.** První ze zmíněných hyperparametrů ovlivňuje o kolik se sníží rychlost učení, v tomto případě se jedná o konstantu, kterou je vynásobená původní rychlost. Znamená to tedy, že hodnota faktoru poklesu rychlosti učení se může pohybovat v hodnotách od 0 do 1, přičemž 1 znamená zachování původní rychlosti. Druhý hyperparametr pak určuje, jak často se rychlost učení bude snižovat. Hodnota periody rychlosti učení říká algoritmu trénování, že by mělo dojít k úpravě rychlosti učení každých „x“ epoch. Pokud bude tedy faktor rychlosti učení nastaven na hodnotě 0,5 a perioda poklesu rychlosti učení na hodnotě 25, pak se rychlost učení sníží každých 25 epoch na polovinu. Vliv rychlosti učení má pak za následek již zmíněné efekty průběhu učení.

**L2 Regularizace** je hyperparametr, jehož účelem je zabránit příliš vysokým hodnotám vah penalizací velikosti vah neuronů, a tím předejít přetrénování sítě. Konkrétně L2 regularizace penalizuje velké váhy tím, že přidává hodnotu součinu hyperparametru a druhé mocniny hodnoty váhy do ztrátové funkce. Hyperparametr může dále přispět ke stabilitě tréninku díky tomu, že omezuje nárůst hodnot vah, což zlepšuje numerickou stabilitu optimalizačního procesu. Hodnota je opět volena empiricky během experimentování s trénováním sítě a bývá naladěna na konkrétní dataset a problém. Obecně vyšší hodnota znamená vyšší snahu opravit špatné váhy, což vede k horší konvergenci, ale i menší náchylnosti k přetrénování.

Hyperparametr **Validační data** určuje, jaká data se použijí pro validaci v průběhu tréninku sítě. Pomocí validačních dat se v průběhu trénování sítě sleduje a vyhodnocuje výkon a přesnost modelu. Z průběhu validace lze přibližně identifikovat, zda se síť dobře generalizuje na nová data či nikoliv, a pomocí toho určit další postup v úpravě hyperparametrů. Určení velikosti validačního datasetu a jeho funkce byla vysvětlena v kapitole 6.2. S tímto hyperparametrem souvisí další hyperparametr **validační frekvence**. Ten určuje, jak často budou data ověřována na validačních datech, přesněji říká, že dojde k validaci každých „x“ epoch.

Mezi další již vedlejší hyperparametry pak patří **výpis**, **grafy** a **prováděcí prostředí.** Pomocí výpisu je možné vypisovat na konzoli informace o tréninku během procesu trénování. Hyperparametr je možno nastavit buď jako vypnutý, nebo jako základní či rozšířený výstup, kde jsou kromě počtu epoch a hodnoty ztrátové funkce i aktuální hodnoty metrik. Hyperparametr grafy slouží k zobrazení grafů a vizualizací během tréninku. Umožňuje nastavení zobrazení ztrátové funkce, přesnosti nebo průběhu učení a validace. Jedná se tedy o užitečný parametr, díky kterému je možné diagnostikovat problémy při učení. Poslední hyperparametr prováděcí prostředí určuje, na jakém prostředí bude síť trénována – zda CPU, GPU nebo paralelně ve více prostředích najednou. Obecně je pro trénování sítí lepší grafický procesor, který je optimalizovaný pro paralelní výpočty a může tak efektivně provádět velké množství operací najednou. Použití GPU místo CPU může tedy značně urychlit proces trénování. Další možností, pokud není k dispozici GPU a je potřeba urychlit trénování sítě, je využití výkonných GPU a clusterů přes cloud.

# 8 SESTAVENÍ RNN PRO ODHAD SMĚROVÉ ÚCHYLKY

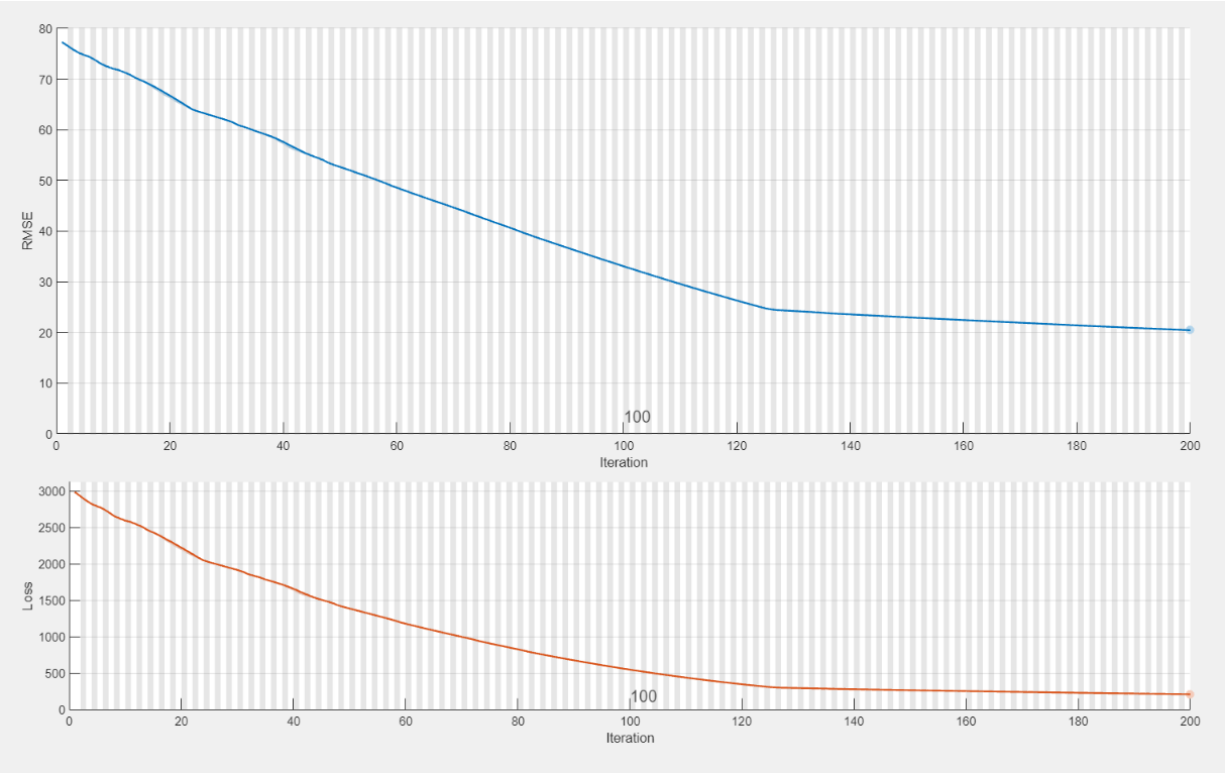
Pro vyzkoušení, zda se bude síť učit a jaký bude průběh samotného procesu učení, je využito základní architektury pro regresi pomocí RNN. Je vybráno a použito 10 vstupních proměnných a 1 výstupní proměnná – rychlost v podélném směru x, jelikož jak bylo avizováno dříve, pro jednodušší sestavení sítě bude nejdříve odhadován pouze 1 výstup. Dále byly zvoleny vybrané hyperparametry viz *tab. 2*.

*Tab. 2* Hodnoty zvolených hyperparametrů

| Hyperparametr | Hodnota |
|---|---|
| SolverName | adam |
| MaxEpochs | 200 |
| GradientThreshold | 1 |
| InitialLearnRate | 0,005 |
| LearnRateSchedule | Piecewise |
| LearnRateDropPeriod | 125 |
| LearnRateDropFactor | 0,2 |
| Verbose | 0 |

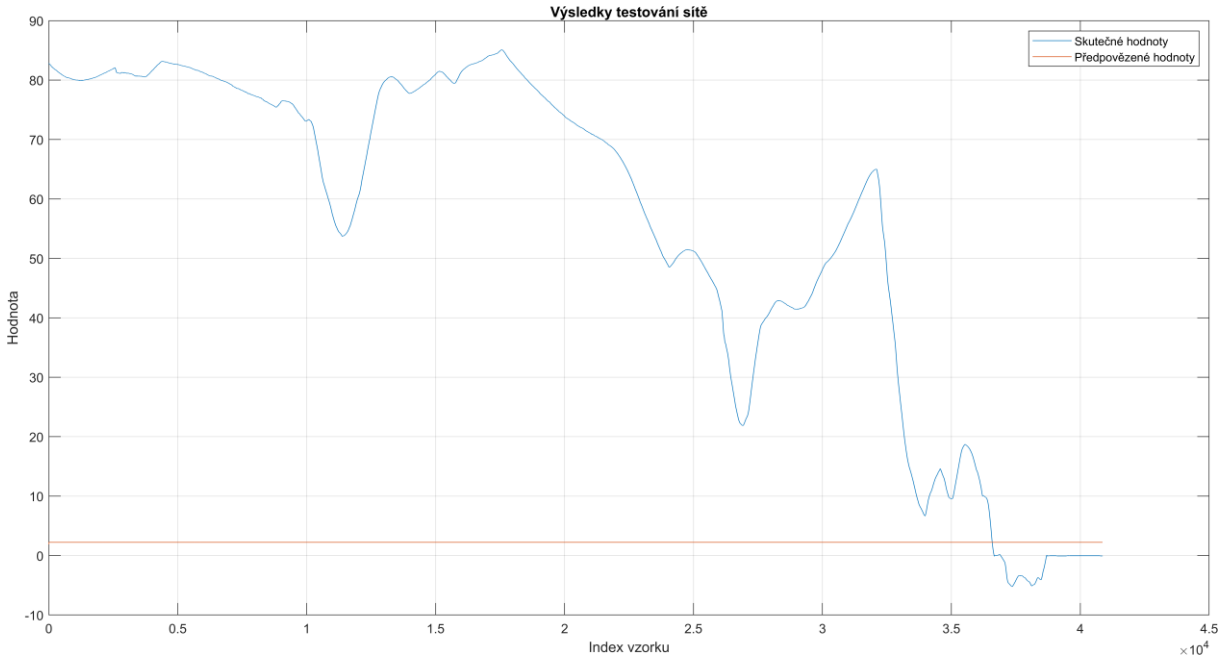
První spuštění trénování sítě však nedopadlo úspěšně a trénování vůbec neproběhlo. Objevil se totiž problém se špatnou dimenzí dat. Vstupní data byla naformátována do matice jako 2D data, ovšem RNN vyžadovala hodnoty dat ve 3D. Hodnoty tedy musely být přeformátovány z matice do pole buněk (cell array), kde jim byla přidána hodnota třetí dimenze 1. Správné rozměry vstupních dat lze tedy specifikovat tak, že vstupní data by měla být pole buněk o rozměrech $1 \times N$, kde $N$ je počet sekvencí. Každý prvek v poli buněk by pak měl být maticí o rozměrech $n \times T$, kde $n$ je počet vstupních (naměřených) proměnných a $T$ je počet časových kroků v sekvenci. Po tomto kroku a zformátování dat bylo již možné trénování sítě spustit. Průběh prvního procesu trénování je na *obr. 31*.

Odmocnina ze střední kvadratické chyby (RMSE) je směrodatná odchylka reziduí. Rezidua jsou měřítkem toho, jak daleko se nacházejí datové body od regresní přímky. RMSE určuje, jak jsou tato rezidua rozptýlena. Jinými slovy udává, jak jsou data soustředěna kolem přímky největší shody. Ztrátová funkce je mírou toho, jak dobře si predikční model vede, pokud jde o schopnost předpovědět očekávaný výsledek. Lze konstatovat, že na první pokus proces trénování nevypadá vůbec špatně. Jelikož RMSE i ztrátová funkce po celou dobu trénování klesají, síť se po celou dobu učí, a to je chtěný výsledek. Nyní je potřeba síť otestovat a zhodnotit ji pomocí testovacích dat, aby se dalo stanovit, zda se jedná o síť s dobrými výsledky.




*Obr. 31* Průběh trénování sítě při prvním pokusu; na horním grafu je střední kvadratická chyba, na spodním grafu je ztrátová funkce

Na *obr. 32* je výsledek natrénované sítě. Modrá křivka jsou skutečné hodnoty, které má síť předpovědět a červená křivka jsou hodnoty, které síť předpověděla. Je patrné, že tato natrénovaná síť je nepoužitelná. Je však potřeba zjistit kde je problém. Zda je síť přetrénovaná, málo natrénovaná, jsou špatně upravená vstupní data nebo zda je síť pouze špatně otestována.




*Obr. 32* Testování první verze natrénované sítě

Pro zanalyzování problému byla použita stejná architektura sítě i stejné zpracování dat, ovšem pro jinou výstupní veličinu, tentokrát již pro směrovou úchylku. Výsledek byl podobný, síť opět predikovala pouze skoro konstantu s tím rozdílem, že zde již byly v některých částech malé náznaky vrcholků.

Vzhledem k problémům s učením neuronové sítě bylo rozhodnuto o normalizaci vstupních dat, což by mělo pomoci k lepšímu přizpůsobení se neuronové sítě na tréninková data, a tedy i k lepším výsledkům. Tento krok nakonec pomohl k tomu, že se síť dokázala učit a začala již napodobovat průběh odhadované veličiny, i když zatím pouze hrubým způsobem. Nyní bylo tedy potřeba neuronovou síť doladit do co nejlepšího stavu, kdy bude kvalitně odhadovat průběh i absolutní hodnoty chtěné veličiny.

## 8.1 ARCHITEKTURA NEURONOVÉ SÍTĚ PRO ODHAD SMĚROVÉ ÚCHYLKY

První odhadovanou veličinou této práce je směrová úchylka. V úmyslu je tedy vytvořit rekurentní neuronovou síť s 1 výstupní a 10 vstupními veličinami. Pro začátek a ověření funkčnosti samotné sítě je sestavena základní rekurentní neuronová síť s následnou architekturou:

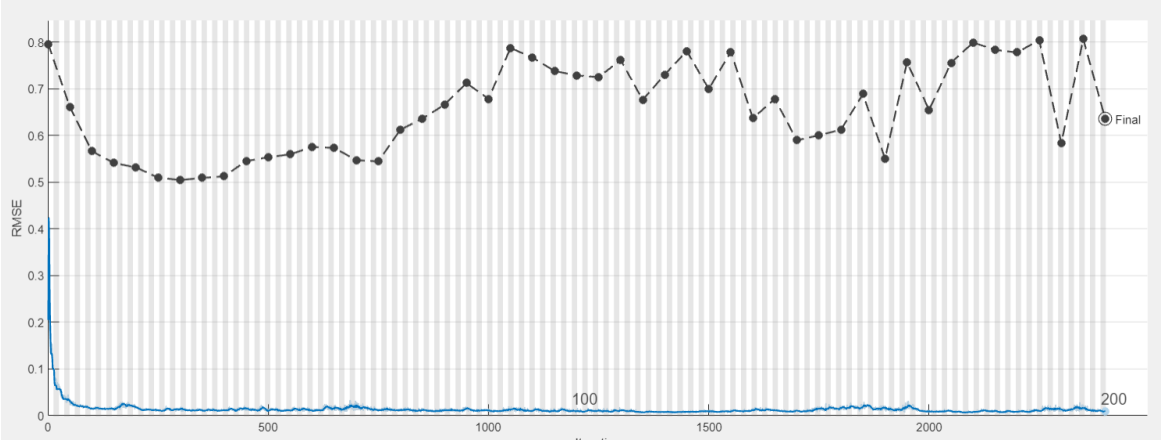
- vstupní vrstva,
- GRU vrstva,
- ReLu aktivační funkce,
- plně propojená výstupní vrstva,
- regresní vrstva.

Takto sestavená síť byla doplněna hyperparametry viz *tab. 3*. Vstupní data byla normalizována a rozdělena do sekvencí po 100 datových jednotkách.

*Tab. 3* Hodnoty zvolených hyperparametrů pro první pokus RNN odhadu směrové úchylky

| Hyperparametr | Hodnota |
|---|---|
| SolverName | adam |
| MaxEpochs | 200 |
| GradientThreshold | 1 |
| InitialLearnRate | 0,01 |
| LearnRateDropPeriod | 100 |
| LearnRateDropFactor | 1 |

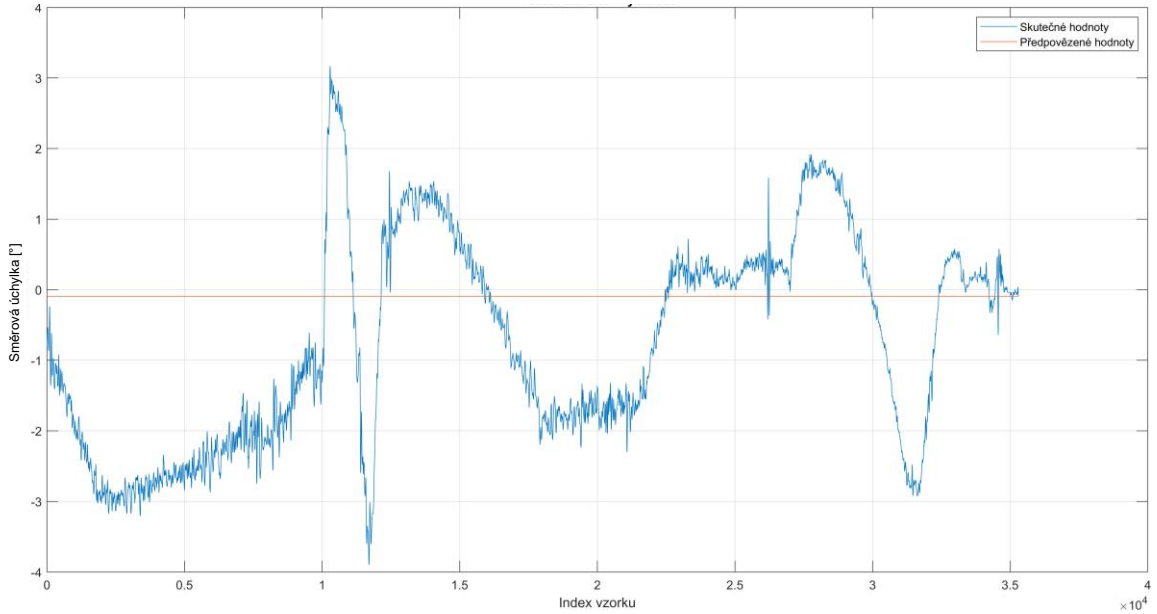
I když samotné ověření na testovacích datech na první pokus trénování nevypadalo vůbec špatně, průběh validace byl již horší.




*Obr. 33* Průběh trénování a validace první sestavené RNN pro odhad směrové úchylky

Na *obr. 33* je modře zobrazen průběh trénování a černě průběh validace, přičemž validace byla nastavena na každých 50 iterací.

Z toho, jak vypadají jednotlivé průběhy, můžeme předpokládat, že je síť přetrénovaná. Je zde totiž vidět, že síť dobře konverguje a rychle se natrénuje, avšak validační data naopak nekonvergují vůbec. Ideální průběh by měl být takový, že bude postupně konvergovat jak proces trénování sítě, tak validace, přičemž by se obě konečné chybové hodnoty měly po natrénování sítě shodovat. Takového ideálního stavu se však v praxi dosahuje těžko. Důležité je tedy snažit se docílit alespoň první části ideálního stavu, tedy postupné konvergence validačních dat i průběhu trénování.




*Obr. 34* Odhadované hodnoty směrové úchylky po úpravě počáteční rychlosti trénování

Odstranit jev přetrénování sítě se dá vícero způsoby. Mezi ten nejjednodušší patří zrychlit učení sítě, aby se síť nestihla příliš přizpůsobit na trénovací data. Hodnota počáteční rychlosti učení 0,01 byla tedy změněna na hodnotu 0,1. I když se rozdíl mezi finální chybou odhadu a natrénované sítě zmenšil, s touto počáteční rychlostí učení se síť učila tak rychle, že nedokázala dokonvergovat a předpovídala opět pouze konstantu viz *obr. 34*.

Je tedy jasné, že zvyšovat počáteční rychlost učení není vždy správné řešení. Hodnota počáteční rychlosti učení byla tedy vrácena na 0,01. Do hyperparametrů však byl přidán hyperparametr shuffle, který zamíchá dle nastavené specifikace trénovací a validační data. To má sice za následek horší schopnost konvergence trénování sítě, ale zároveň zlepšení konvergence validace, jelikož se síť trénuje na různě se prohazujících a měnících datech. Síť se tedy tolik nepřizpůsobí jednomu typickému průběhu dat. Dalším způsobem, jak předejít přetrénování a zlepšit obecnou generalizaci sítě na nová data, je použití hyperparametru L2Regularization. Tento hyperparametr byl tedy přidán s hodnotou 0,001. I když použití těchto hyperparametrů zlepšilo konvergenci trénování i validace, stále se nejednalo o zásadní zlepšení. Pro potřebné zlepšení bylo tedy nutné najít a určit vhodnější hodnoty a konfigurace jednotlivých hyperparamerů.

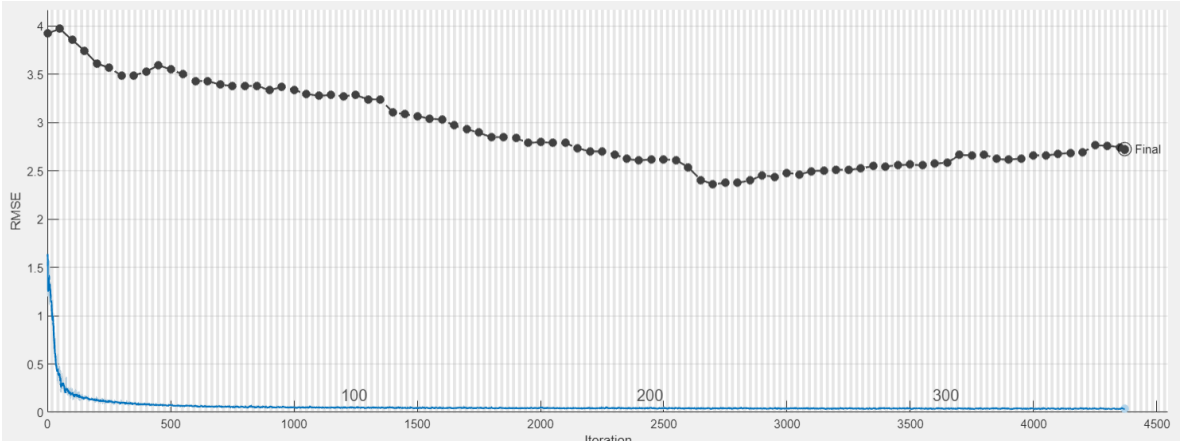
### 8.1.1 VOLBA HODNOT HYPERPARAMETRŮ

Jelikož se hodnota u vícero parametrů volí spíše empiricky a záleží na řešeném problému i velikosti datasetu, je relativně těžké zvolit správné počáteční hyperparametry tak, aby síť fungovala správně. Je tedy dobré inspirovat se u jiné neuronové sítě, která se zabývá podobným problémem. Neuronová síť pro predikci směrové úchylky, respektive počáteční hodnoty hyperparametrů, byly inspirovány prací, kterou vytvořili Hermansdorfer a spol. [38]. Práce se zabývá vytvořením neuronové sítě pro odhad dynamických veličin pro autonomní vozidlo v dalším časovém kroku. Základní podobnosti pro inspiraci zmíněnou prací jsou vstup i výstup dynamických veličin měřených v čase, kdy se jedná o sekvenční data. Dále je v práci vytvořena RNN síť, která je i předmětem této práce. Na druhou stranu účely použití těchto sítí se liší, a i velikosti datasetů a konkrétní veličiny jsou různé. Inspirace počátečními hodnotami hyperparametrů je tedy možná, ale rozhodně budou muset být hodnoty upraveny tak, aby seděly na konkrétní problém a dataset. Zvolené počáteční hodnoty pro trénink jsou v *tab. 4*.

*Tab. 4* Zvolené počáteční hodnoty hyperparametrů

| Hyperparametr | Hodnota |
|---|---|
| SolverName | adam |
| MaxEpochs | 1000 |
| GradientThreshold | 1 |
| Shuffle | every epoch |
| InitialLearnRate | 0,001 |
| LearnRateDropPeriod | 100 |
| LearnRateDropFactor | 1 |
| L2Regularization | 0,001 |
| ValidationFrequency | 50 |

Hodnota vedlejšího hyperparametru výpis byla zvolena „1" po celou dobu úpravy hyperparametrů, aby se vypisovaly informace trénování do konzole. Dále, protože byla možnost využít GPU, bylo prováděcí prostředí nastaveno na GPU pro zrychlení trénování sítě. Architektura sítě byla nezměněna a zůstal i počet skrytých neuronů a délka sekvencí. Průběh při tomto nastavení je na *obr. 35*.

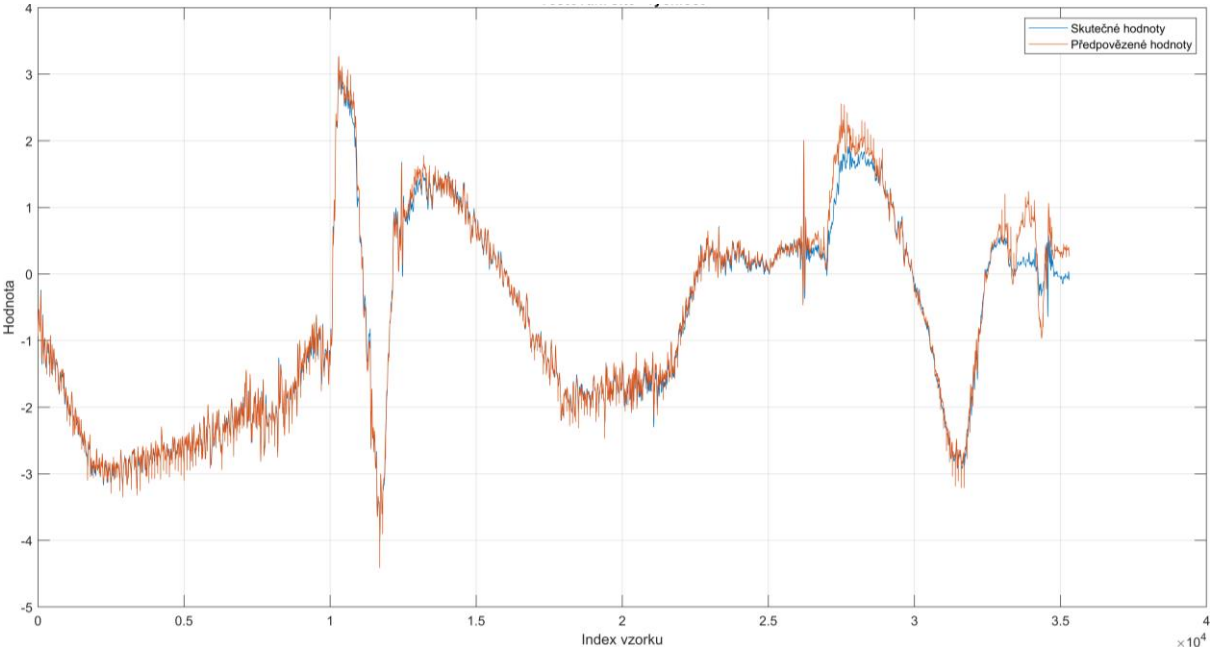



*Obr. 35* Průběh trénování a validace se všemi zadanými hyperparametry

Z průběhu validace a trénování je vidět, že trénink sítě se sice ustálil a pěkně konverguje, ale to se nedá říct o datech validačních, která jsou na tom ještě hůř než v předchozím pokusu a nastavení. Dá se odhadovat, že je tedy síť stále přetrénovaná a málo generalizovaná na nová data. Průběh odhadovaných dat této sítě je na *obr. 36*. Zajímavé je, že přestože validační data nemají dobré výsledky, samotný odhad dat je na tom vcelku dobře, což je pozoruhodný jev, protože správně by měla predikce dat testovací části datasetu reflektovat konečnou hodnotu přesnosti validace, tedy predikce by měla být spíše špatná, což úplně neodpovídá. Na druhou stranu i když síť udělá relativně správnou predikci na testovacích datech, může se jednat o „náhodu", kdy přestože je síť málo generalizovaná, může být správně natrénovaná na predikovaná data. Snaha bude tedy najít nastavení s lepšími výsledky validace.

Postupnými změnami hodnot parametrů bylo iteračně dosaženo natrénování RNN sítě s poměrně dobrými výsledky na testovacích datech. Po základním vyladění sítě a několika trénovacích relacích bylo zjištěno, že bude potřeba vstupní data ořezat tak, aby obsahovala pouze dynamické jevy z průběhu jízdy a neobsahovala již např. popojíždění vozidla na parkovišti či parkování do garáže. Jedná se totiž o část datasetu, kvůli kterému docházelo ke špatnému vyhodnocení přesnosti natrénované sítě. Část způsobující špatné vyhodnocení je na *obr. 36* v hodnotách indexu od $3 \cdot 10^4$ po konec. V těchto hodnotách totiž síť v žádné z relací nebyla správně schopna predikovat hodnoty. To je způsobené tím, že se síť učila na dynamických jevech z jízdy vozidla, zatímco data z parkování jsou naměřena pouze na konci cesty, kterou obsahovala dle rozdělení viz kap. 6.2 pouze testovací sada dat. Jelikož chyba sítě byla počítána na základě metriky RMSE, která je nejvíce náchylná právě na největší odchylky predikovaných dat od dat reálných, většinu chyby způsobovala zmíněná problémová část a zbytek sítě se tím pádem v hodnocení „ztratil".

Stalo se tedy, že i když v průběhu jízdy síť 1 odhadovala hodnoty lépe než síť 2, byla vyhodnocena jako horší, protože měla horší výsledky v problémové části, na které je metrika více náchylná. Po odstranění problémové části byl však problém vyřešen a síť mohla být již adekvátně hodnocena metrikou RMSE.




*Obr. 36* Průběh skutečných a předpovídaných hodnot směrové úchylky po natrénování sítě se všemi zadanými hyperparametry

Postupně bylo vyzkoušeno více variant architektury sítě s různým počtem vrstev i aktivačních funkcí. Byla vyzkoušena jak vrstva LSTM, tak GRU. Jelikož se ale v tomto případě nejedná o dlouhé sekvence dat, maximálně v řádech 1000 jednotek dat, má RNN síť lepší výsledky s vrstvou GRU než LSTM. Postupně byly vyzkoušeny 3 základní druhy aktivačních funkcí mezi vrstvami – ReLU vrstva, sigmoidní vrstva a vrstva tanh. Nejhůře performovala vrstva tanh, která transformuje výstup z předchozí vrstvy na hodnoty v rozsahu (-1,1). Zde se často stávalo, že výstup byl „ořezaný“ od nějaké konstantní hodnoty směrem nahoru a vyšší hodnoty než konstanta se tedy již v predikci neobjevily. Sigmoidní vrstva měla ze všech vrstev nejlepší výsledky. Problém nastal při architektuře sítě s vyšším počtem vrstev, kdy začalo docházet k jevu mizejících gradientů. Tento problém však řeší ReLU vrstva, která není tolik náchylná na mizející gradienty a dala se tedy využít i při složitější architektuře či vyšší rychlosti učení. I když mělo použití ReLu vrstvy přijatelné výsledky, nebyly tak kvalitní jako s aktivační sigmoidní funkcí.

Byly vyzkoušeny architektury, kde bylo za sebou připojeno více plně propojených vrstev i více vrstev GRU. Dále byl vyzkoušen různý počet neuronů ve všech vrstvách, ale nakonec dávala nejlepší výsledky jednodušší architektura sítě. Aby bylo dosaženo stejně kvalitní predikce jako predikce od jednoduché sítě, musela být síť vcelku rozsáhlá – obsahovat několik skrytých i GRU vrstev a několik stovek neuronů v každé vrstvě. Ke všemu se pak taková síť i dlouho trénovala, což je oproti jednoduché síti se stejnou kvalitou predikce a výrazně rychlejším časem

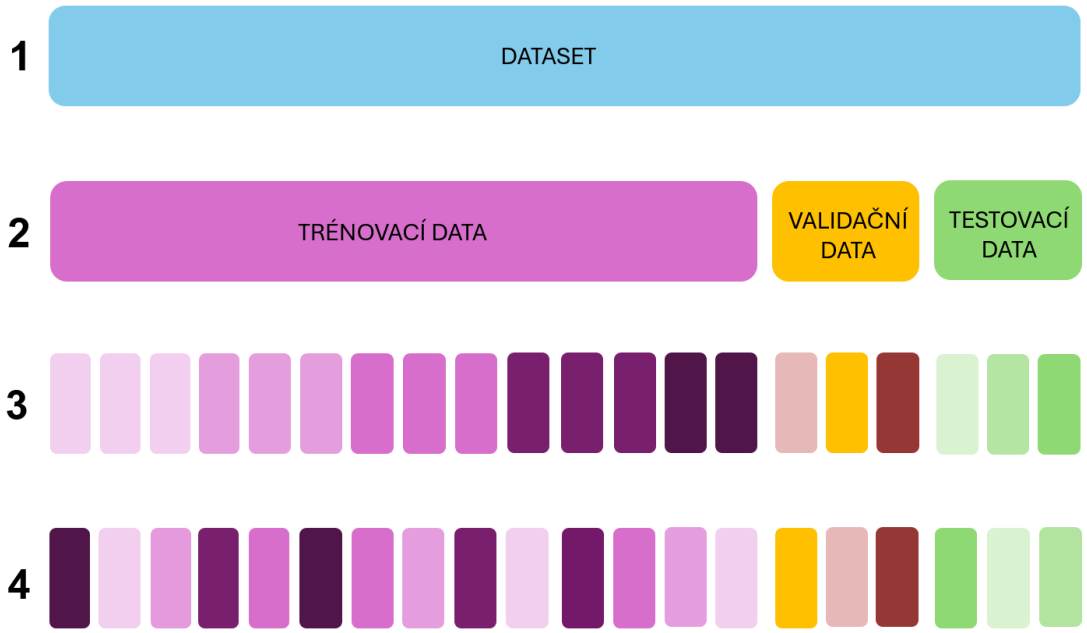
natrénování neefektivní. Výsledkem iteračního procesu hledání vhodné architektury RNN tedy je sestavení té nejjednodušší možné sítě bez jakékoliv aktivační funkce – přesné složení je následující:

- vstupní vrstva,
- GRU vrstva,
- plně propojená vrstva,
- výstupní regresní vrstva.

Vstupní vrstva obsahuje 11 neuronů kvůli 11 vstupním veličinám, GRU vrstva obsahuje 250 neuronů, přičemž k této hodnotě bylo postupně doiterováno tím způsobem, že se postupně zvyšoval počet neuronů, dokud se zvyšovala přesnost odhadu sítě, ale zároveň nedocházelo k přetrénování, 250 neuronů tedy byla hranice. Plně propojená vrstva pak obsahuje 1 neuron pro 1 výstupní veličinu.

Jednotlivé hyperparametry byly nastaveny následovně:

- Jako **optimalizátor** (solver) byl vybrán algoritmus Adam, jelikož ze všech řešičů ve všech případech nejlépe vedl ke konvergenci sítě.
- **Gradient Threshold** byl zachován na hodnotě „1“ jako kompromis mezi stabilitou sítě a rychlostí učení.
- **Shuffle** byl nastaven na hodnotu „every-epoch“. To výrazně zlepšilo generalizaci sítě a snížilo náchylnost na přetrénování. Jak již bylo zmíněno, hyperparametr shuffle totiž slouží k zamíchání jak vstupních, tak validačních dat. Důležité je však zmínit, že data jsou sice zamíchána, ale pouze po celých sekvencích, a ne po jednotlivých datových záznamech.

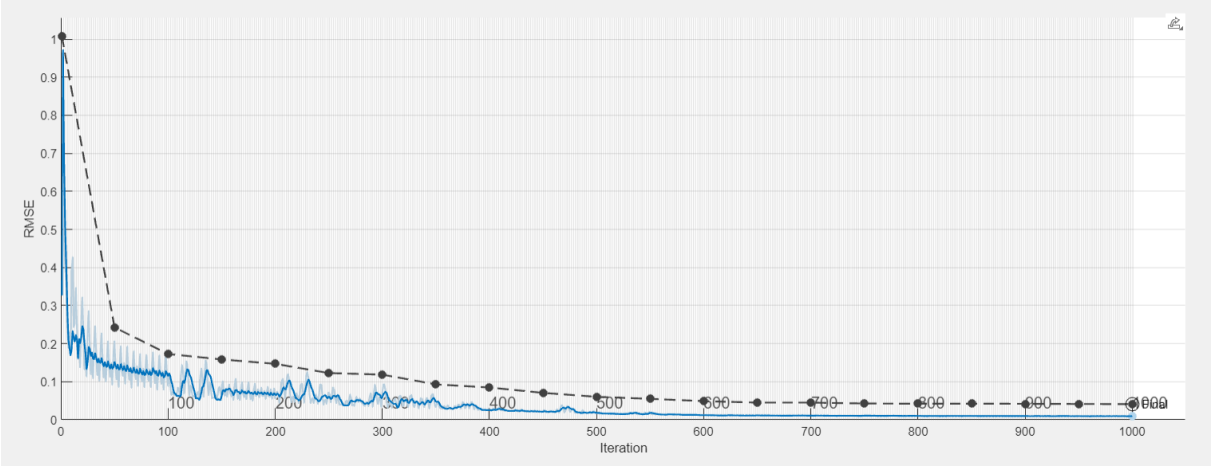



*Obr. 37* Schéma rozdělení a zamíchání datasetu; 1 – naměřený dataset; 2 – rozdělený dataset dle zvoleného poměru; 3 – rozdělené části datasetu na sekvence; 4 – zamíchané sekvence

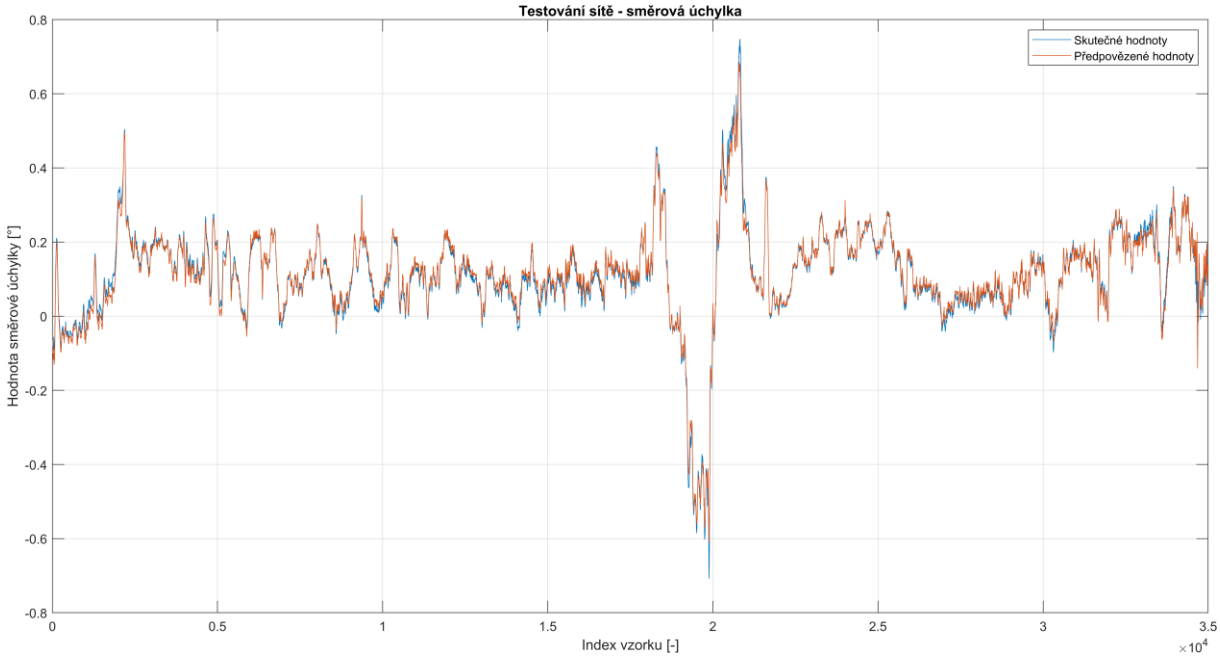
Všechna data jsou tedy rozdělena do sekvencí po x záznamech, přičemž jednotlivé sekvence jsou mezi sebou zamíchány – pro lepší názornost je proces zobrazen na *obr. 37*. První řádek znázorňuje naměřený dataset, druhý řádek rozdělený dataset na trénovací, validační a testovací data (v této práci v poměru 70:15:15). Třetí řádek znázorňuje části dat rozdělené na jednotlivé sekvence, přičemž pro názornost jdou barvy postupně od nejsvětlejší po nejtmavší. Čtvrtý řádek pak znázorňuje již zamíchané sekvence mezi sebou, kdy barvy již nejdou popořadě. Každá sekvence tedy obsahuje x datových jednotek, které však nejsou v sekvenci zamíchány a jdou v čase za sebou.
Míchání pouze po sekvencích je důležité pro zachování dynamických jevů v naměřených datech. V případě míchání po jednotlivých datových záznamech by byly dynamické jevy rozloženy a zároveň by ztratilo podstatu, že jde o sekvenční data měřená v čase. S tím souvisí i správná volba délky sekvence. Ta nemůže být moc krátká, protože by z ní síť nedokázala vyčíst žádný dynamický jev a vzorec v datech. Příliš dlouhá sekvence není vyloženě v ničem špatná, pouze zbytečně zvyšuje výpočetní náročnost a tím i čas tréninku, čímž se snižuje efektivita. Obecně se však doporučuje najít optimální hodnotu, která by měla být taková, která reflektuje délku sledovaných jevů. V této práci se nejlepší výsledky pohybovaly v rozmezí délky sekvencí 800 až 1000 datových jednotek na sekvenci. Nakonec byla zvolena hodnota 1000 datových jednotek na sekvenci. Při vyšších hodnotách se již znatelně snižovala rychlost učení, tím i rychlost konvergence sítě, a i výsledné predikce sítě byly horší.

- **Initial Learn Rate** byl nastaven na hodnotu 0,003. Pro tento typ úlohy a dat měly nejlepší výsledky hodnoty v jednotkách tisícin. O řád vyšší hodnoty již nekonvergovaly a odhadovaly pouze konstantu viz *obr. 34*. Nižší řády způsobovaly, že se síť učila příliš pomalu a opět buď vůbec nebo pouze málo konvergovala.
- **Learn Rate Drop Factor** byl nastaven na „0,5“ a **Learn Rate Drop Period** na „100“. Každých 100 epoch se tedy rychlost učení snížila na polovinu. Postupné snižování rychlosti zajistilo, že se síť nepřetrénovala. Hodnota snižování byla nastavena tak, aby snižování bylo dostatečně k zajištění generalizace sítě, ale zároveň aby byla síť schopna konvergovat.
- Vedlejší hyperparametry zůstaly stejné, jak již bylo zmíněno výše, kdy **Verbose** byl zvolen na hodnotu „1“, byla zvolena validační data pro ověření s frekvencí 50 epoch a **Execution Environment** byl z důvodu zrychlení výpočtu zvolen jako GPU.

Průběh tréninku a validace takto vytvořené RNN je na *obr. 38*. Z průběhu je vidět, že síť postupně konverguje jak v tréninku, tak validaci. Průběh tréninku je opět zobrazen modře a průběh validace černě. Ve výsledku se validace přibližuje tréninku, což značí, že je síť dobře generalizována na nová data. Jedná se vcelku o ideální průběh. Na *obr. 39* je následně síť otestována na testovacích datech. Modře jsou zobrazeny hodnoty skutečné a červeně předpovězené. Síť odhaduje dobře jak průběh dat, tak i jejich absolutní hodnotu. RMSE natrénované sítě je 0,011233, což je v porovnání s průměrnou hodnotou naměřených hodnot úhlu směrové úchylky, která je 0,1138 °, přijatelná hodnota.




*Obr. 38* Průběh trénování a validace vytvořené RNN sítě pro odhad směrové úchylky




*Obr. 39* Ověření natrénované sítě pro odhad směrové úchylky na testovacích datech

# 9 OVĚŘENÍ ZÁVISLOSTI PŘESNOST SÍTĚ NA VYBRANÉ VSTUPNÍ VELIČINY

Dalším krokem je ověřit závislost přesnosti natrénované sítě na vybraných vstupních veličinách. Jde o to, aby se analyzovalo, na které vstupní veličiny je třeba klást důraz při měření, aby měla síť kvalitnější predikci a zároveň aby se případně mohly vymazat vstupy, na kterých by síť byla nezávislá.

Postupně tedy bylo vyzkoušeno natrénování sítě bez jednotlivých skupin vstupů viz *tab. 5*, kdy pro každý vstup byla síť natrénována pětkrát. Jako hodnocení kvality sítě bylo u každého pokusu bráno RMSE, které bylo následně zprůměrováno, přičemž tyto průměrné hodnoty byly pak mezi sebou porovnávány. Skupiny vstupů byly následující:

- vstupy zrychlení ve všech osách (aX, aY, aZ),
- vstup úhlu sklonu vozidla (pitch angle),
- vstup úhlu náklonu vozidla (roll angle),
- vstupy úhlové rychlosti ve všech osách (wX, wY, wZ),
- vstupy rychlostí ve všech osách (vX, vY, vZ),
- vstupy zdvihu tlumičů na všech kolech (dmpFR, dmpFL, dmpRR, dmpRL).

Základní nastavení sítě se všemi vstupními veličinami obsahovalo 11 vstupních hodnot, kdy vstupy z tlumičů na jednotlivých kolech nebyly do základního nastavení zahrnuty. Jediné tyto vstupy byly tedy při testování přidány. Ostatní vstupy byly postupně po skupinách odebrány. Jak již bylo zmíněno, aby se dala považovat hodnota RMSE pro jednotlivá nastavení za relevantní, bylo trénování v daném nastavení zopakováno pětkrát.

*Tab. 5* Přehled výsledků trénovaných sítí s různým nastavením

| Vstupní parametry | 1. test | 1. validace | 2. test | 2. validace | 3. test | 3. validace | 4. test | 4. validace | 5. test | 5. validace | Test průměr | Validace průměr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| všechny | 0,014 | 0,061 | 0,011 | 0,059 | 0,023 | 0,081 | 0,014 | 0,044 | 0,015 | 0,065 | 0,0154 | 0,0618 |
| všechny - aX, aY, aZ | 0,016 | 0,058 | 0,016 | 0,049 | 0,012 | 0,062 | 0,011 | 0,055 | 0,014 | 0,057 | 0,0138 | 0,0562 |
| všechny - pitch | 0,015 | 0,080 | 0,020 | 0,056 | 0,022 | 0,090 | 0,014 | 0,042 | 0,014 | 0,055 | 0,0169 | 0,0649 |
| všechny - roll | 0,016 | 0,053 | 0,011 | 0,062 | 0,015 | 0,072 | 0,014 | 0,064 | 0,013 | 0,057 | 0,0138 | 0,0614 |
| všechny - wX, wY, wZ | 0,017 | 0,034 | 0,015 | 0,027 | 0,009 | 0,027 | 0,010 | 0,028 | 0,014 | 0,039 | 0,0129 | 0,0310 |
| všechny - vX, vY, vZ | 0,092 | 0,161 | 0,093 | 0,154 | 0,085 | 0,146 | 0,179 | 0,191 | 0,091 | 0,187 | 0,1080 | 0,1679 |
| všechny + dmp | 0,084 | 0,177 | 0,086 | 0,365 | 0,073 | 0,261 | 0,084 | 0,219 | 0,071 | 0,171 | 0,0795 | 0,2385 |
| vX, vY, vZ + pitch + roll | 0,009 | 0,029 | 0,011 | 0,031 | 0,010 | 0,038 | 0,008 | 0,028 | 0,010 | 0,036 | 0,0097 | 0,0324 |
| vX, vY, vZ | 0,010 | 0,018 | 0,012 | 0,040 | 0,013 | 0,025 | 0,008 | 0,018 | 0,010 | 0,025 | 0,0104 | 0,0252 |

Pro porovnání kvality jednotlivých sítí se bere v úvahu hodnota RMSE testování natrénované sítě i konečná hodnota RMSE validace. Do porovnávacího kritéria jsou zahrnuty testování i validace z toho důvodu, že se stalo, že přestože síť měla na testovacích datech dobré výsledky, tak na validačních byly špatné a síť nemohla být brána jako dobře generalizovaná na nová data. Je tedy potřebné mít dobré výsledky v obou zmíněných případech. Výsledky tohoto testování jsou v *tab. 5*.

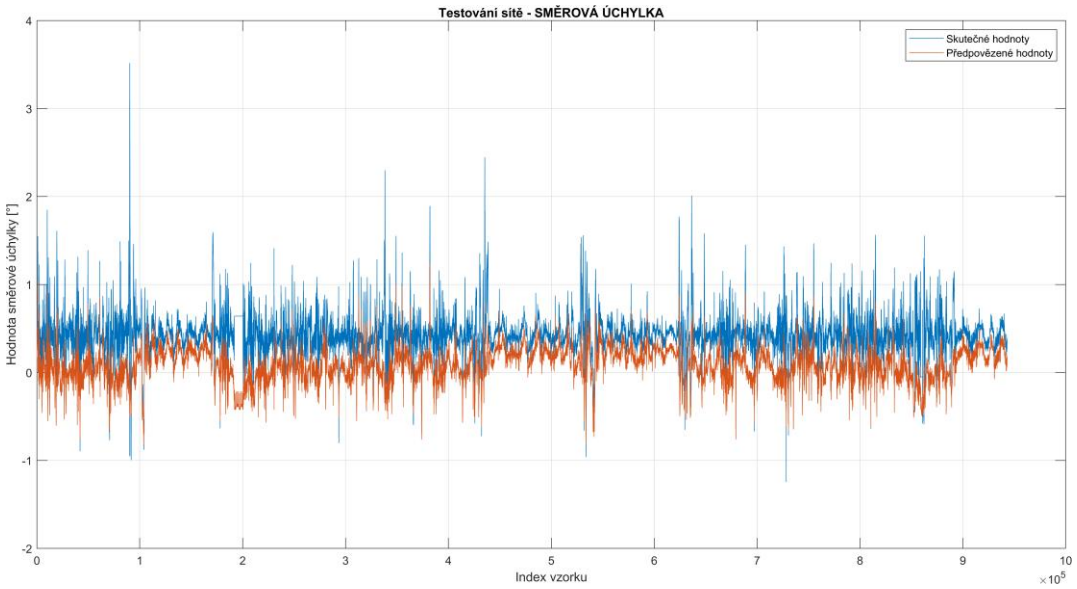
Z *tab. 5* je možné vyčíst, že přidání vstupních veličin zdvihu všech tlumičů (dmp) kvalitu sítě zhoršilo oproti základnímu nastavení se všemi vstupními parametry. Největší vliv na kvalitu sítě mají vstupy rychlostí, které když byly odebrány, zhoršily průměrný RMSE sítě na nejhorší výsledek. Naopak u ostatních vstupů, které byly postupně odebrány byly výsledky srovnatelné a někdy dokonce i lepší než v základním nastavení. Z tohoto důvodu byla do ověření přidána ještě další 2 nastavení. První zahrnovalo vstupy rychlostí ve všech směrech, úhlu sklonu vozidla a úhlu náklonu vozidla. Druhé pak již zahrnuje pouze vstupy rychlostí. Výsledky hovoří jasně, síť má překvapivě lepší výsledky s méně vstupy než v základním nastavení se všemi vstupy. Nejlépe ze všech případů vyšlo nastavení s pouze třemi vstupy – rychlostmi ve všech směrech. Konečné průměrné hodnoty RMSE tohoto nastavení jsou zvýrazněny v *tab. 5* zeleně. Zároveň měl jeden z pokusů trénování tohoto nastavení nejlepší výsledek RMSE ze všech pokusů o natrénování. Tato síť je následně použita jako výsledná natrénovaná síť pro odhad směrové úchylky.

Konečný výsledek nastavení sítě je tedy takový, že byl změněn počet vstupních veličin, kdy do finální verze RNN sítě vstupují pouze 3 veličiny, přičemž architektura a hyperparametry byly zachovány z původního nastavení popsaného v kapitole 8.1.1.

# 10 OTESTOVÁNÍ RNN PRO PREDIKCI SMĚROVÉ ÚCHYLKY

Dále je třeba natrénovanou síť otestovat na jiném datasetu, než který byl použit na natrénování sítě. Pro otestování jsou využita data naměřená stejnou inerciální jednotkou na stejném vozidle, ale z jiné jízdy. Data jsou opět na začátku a konci ořezána tak, aby obsahovala pouze části z jízdy obsahující dynamické jevy. Po ořezání obsahuje dataset 943000 časových kroků, záznam je tedy dlouhý zhruba 157 minut. Jedná se o větší dataset, než na kterém byla síť natrénována.

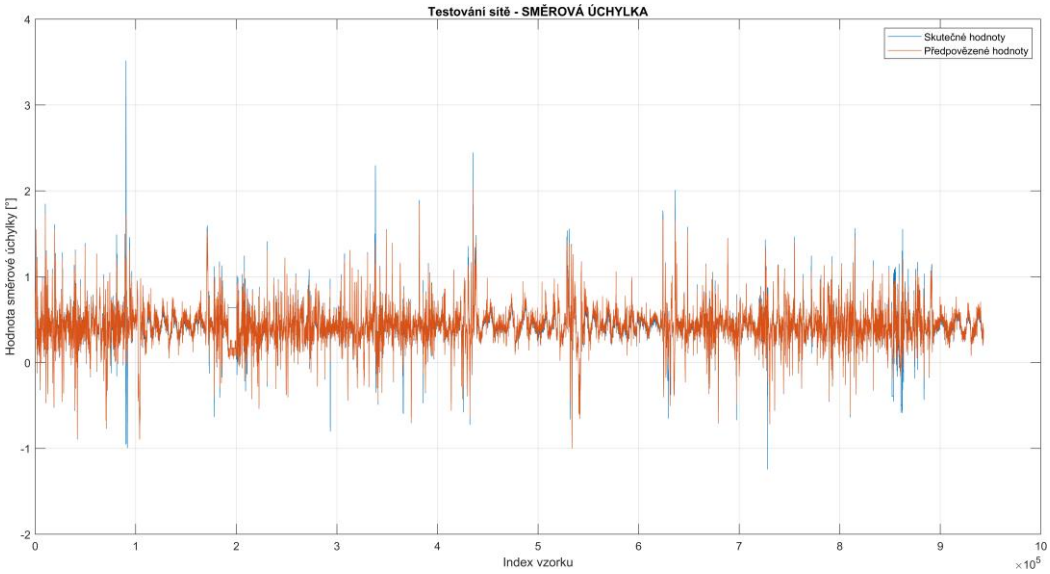
Data z datasetu pro otestování vstupující do vytvořené sítě musí být upravena stejným způsobem jako byla upravena data pro natrénování sítě. To znamená, že jsou nejdříve pomocí dolnopropustného filtru z dat odstraněny nežádoucí frekvence, následně jsou data rozdělena do sekvencí po 1000 datových jednotkách, a nakonec jsou normalizována. Poté stačí poslat vstupy do sítě a ta předpoví výsledek. Výsledek predikce na novém datasetu je na *obr. 40*.



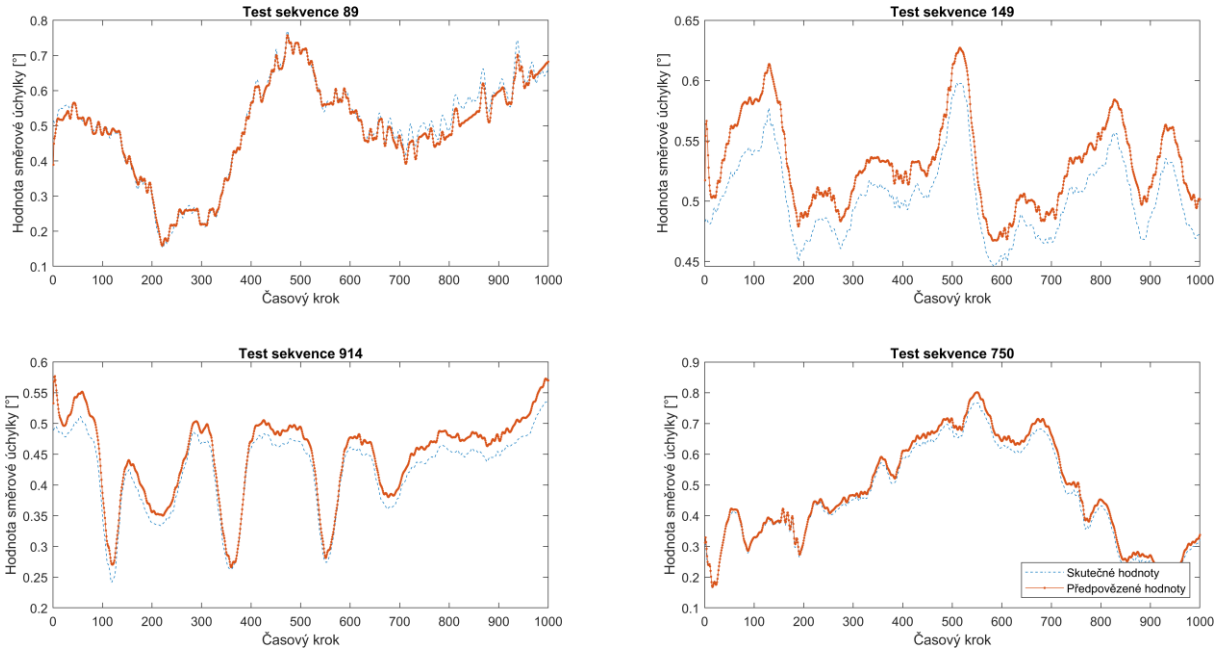

*Obr. 40* První testování neuronové sítě pro odhad směrové úchylky na jiném datasetu

Modře zobrazeny jsou skutečné hodnoty a červeně hodnoty předpovídané. Je vidět, že průběh hodnot sice vypadá dobře, ale absolutní hodnoty jsou posunuty o konstantní hodnotu vůči hodnotám skutečným. Vzhledem k tomu, že jinak průběh vypadá dobře, je nejspíše problém způsobený tím, že síť nemá žádný vstupní parametr, který by jí dodával informaci o správném „výškovém“ posunu hodnot.

Příčina byla nakonec nalezena bez nutnosti přidávání dalších vstupních veličin. Problém byl v normalizaci dat, kvůli které se z dat ztratily informace o „výškovém“ posunu hodnot. Normalizace tedy byla odstraněna ze zpracování dat pro natrénování sítě a stejně tak i ze zpracování testovacích dat. Opětovné testování, tentokrát již bez normalizace, dopadlo viz *obr. 41* výrazně lépe. Opět modře jsou zobrazeny skutečné hodnoty a červeně hodnoty předpovězené. Při tomto testování jsou již v pořádku jak absolutní hodnoty, tak průběh dat. Na *obr. 42* jsou čtyři náhodné sekvence pro detailnější zobrazení rozdílu v průběhu odhadovaných a naměřených dat.




*Obr. 41* Finální testování neuronové sítě pro odhad směrové úchylky



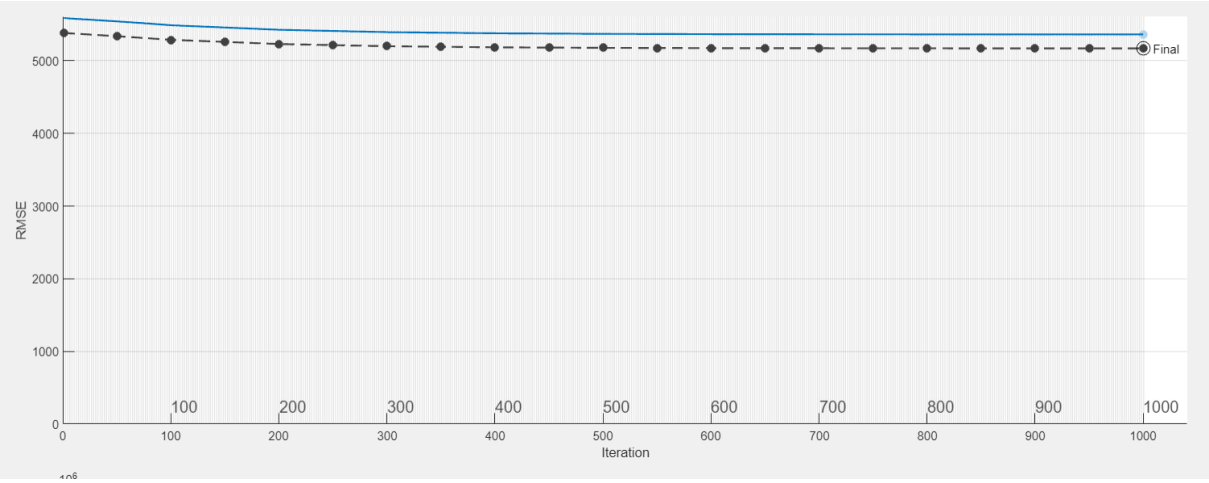

*Obr. 42* Zobrazení náhodných sekvencí z finálního testování neuronové sítě

Z testování je dále vidět, že síť nedokáže úplně přesně předpovědět náhlé výrazné odchylky viz *obr. 41* okolo indexu vzorku $1 \cdot 10^5$. Ovšem většinu zbylých odchylek síť odhadla relativně správně. Další problém při predikci je konstantní hodnota směrové úchylky. Tento jev je na *obr. 41* okolo indexu vzorku $2 \cdot 10^5$. Je vidět, že síť místo konstantní hodnoty předpovídá jemné kmitání na jiné hodnotě. To je nejspíše způsobené tím, že trénovací data neobsahují žádný takový jev, kde by byla konstantní hodnota směrové úchylky a síť proto není na tento jev naučená. Finální hodnota RMSE na testovacích datech je 0,079586, což je vůči průměrné hodnotě směrové úchylky z naměřených dat 0,4289 ° přijatelná hodnota.

# 11 ARCHITEKTURA NEURONOVÉ SÍTĚ PRO ODHAD PODÉLNÝCH SIL NA KOLECH VOZIDLA

Z počátku bylo v plánu, že se vytvoří pouze jedna RNN pro odhad směrové úchylky i podélných sil na kolech. V průběhu trénování se však došlo k závěru, že vzhledem k rozdílným veličinám i absolutním hodnotám veličin je to nepraktické řešení a zbytečně by to zhoršovalo kvalitu predikce všech veličin. Pro odhad podélných sil na kolech byla tedy vytvořena samostatná neuronová síť.

Počáteční hodnoty pro tvorbu neuronové sítě byly přebrány z již vytvořené neuronové sítě pro odhad směrové úchylky. Byly použity úplně stejné hodnoty s cílem rozchodit základ neuronové sítě pro další případné úpravy. Jediný rozdíl byl v tom, že byly přidány vstupy rychlostí z tlumičů a celkem tedy na začátku bylo 15 vstupních veličin. Místo 1 výstupu zde pak byly 4 – podélná síla pro každé kolo zvlášť. Natrénování sítě s těmito počátečními hodnotami však nebylo úspěšné, protože síť se s takovým nastavením nechtěla vůbec učit. Průběh tréninku je na *obr. 43*.

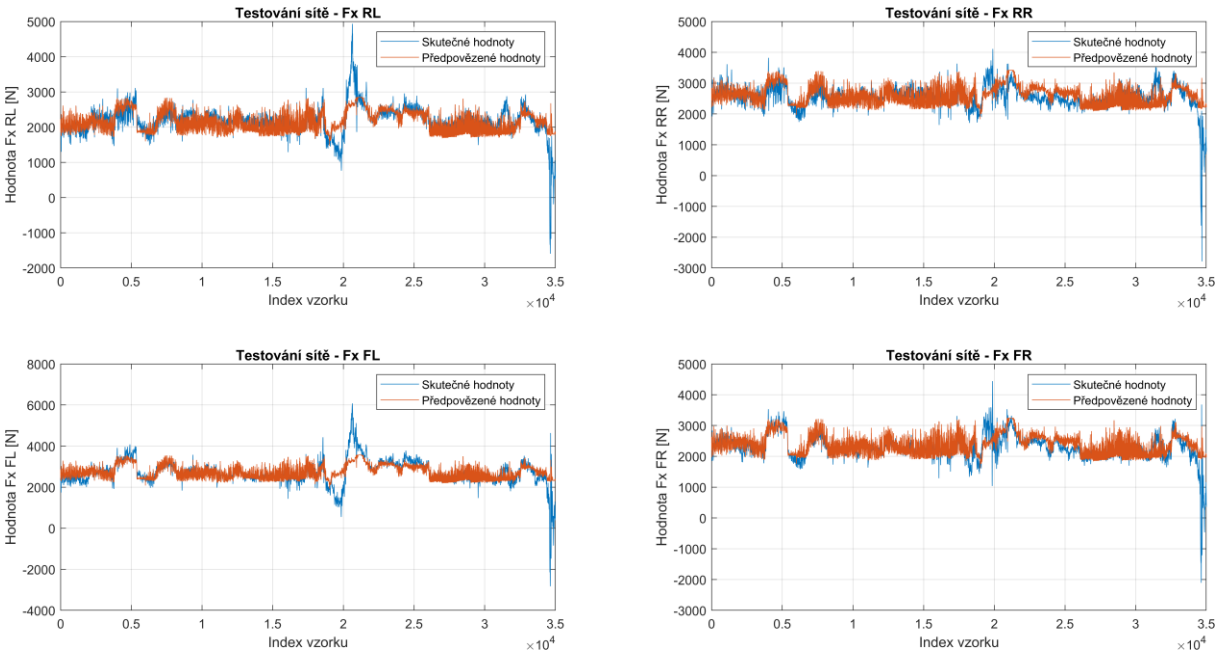



*Obr. 43* Průběh prvního pokusu trénování RNN pro odhad podélných sil na kolech vozidla

Modře je zobrazen průběh procesu trénování a černě průběh validace. Z takového průběhu lze již předpokládat, že síť nebude schopná předpovědět správné hodnoty a předpoví pouze konstantní hodnotu. Jelikož síť nebyla schopna se ani natrénovat, bylo potřeba upravit rychlost trénování. Pomocí dalších pokusů o natrénování bylo zjištěno, že počáteční rychlost trénování byla příliš nízká, a proto se síť nechtěla přizpůsobit. Postupně bylo vyzkoušeno více hodnot počátečních rychlostí, přičemž s hodnotou 0,09 se síť konečně začala alespoň učit. I když se s touto hodnotou síť začala učit, výsledná predikce byla stále konstantní hodnota.

Byly vyzkoušeny různé hodnoty vícero hyperparametrů, ale nic nevedlo ke zlepšení predikce sítě. Nakonec bylo zjištěno, že problém je v tom, že data nejsou normalizována (postup zpracování dat byl také přebrán z první vytvořené sítě). Stačilo tedy přidat do úpravy trénovacích dat normalizaci a síť začala mít predikce, které jsou zobrazeny na *obr. 44*. Už z průběhu se však dá určit, že se nejednalo o příliš dobrý výsledek.

Je tedy opět potřeba najít nejen správnou hodnotu hyperparametrů, ale i optimální architekturu neuronové sítě. Vcelku velký problém zde však zapříčinila vysoká hodnota počáteční rychlosti učení, která způsobovala explodující gradienty, a tedy neschopnost sítě se učit. Jelikož je však vysoká počáteční hodnota nutná k natrénování sítě, musela být proti tomu udělána jiná opatření, jako například snížení počtu neuronů v jednotlivých vrstvách nebo průběžné snižování rychlosti učení po častějších intervalech případně i s nižším koeficientem.




*Obr. 44* První predikce podélných sil na kolech

Opět byly postupně vyzkoušeny různé variace architektur s různými hodnotami hyperparametrů. Byla vyzkoušena i možnost jiného rozložení trénovacích, validačních a testovacích dat. Celý dataset byl postupně rozdělen na první polovinu trénovacích dat, poté byla část validačních dat, následovala část testovacích dat, a nakonec druhá polovina dat trénovacích. Bohužel to ke generalizaci dat nepomohlo, spíše naopak. Po všech vyzkoušených kombinacích různých hodnot hyperparametrů a architektur sítě se nakonec došlo k této stavbě sítě:

- vstupní vrstva,
- GRU vrstva,
- GRU vrstva,
- ReLu aktivační funkce,
- plně propojená vrstva,
- plně propojená výstupní vrstva,
- regresní vrstva.

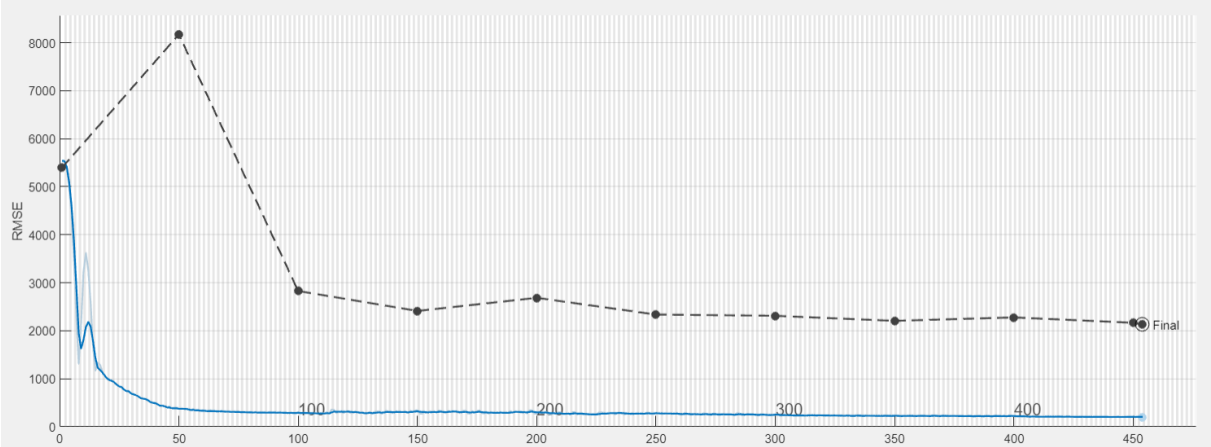
Síť má již zmíněných 15 vstupních veličin a 4 veličiny výstupní. Vstupní vrstva má tedy 15 neuronů a výstupní vrstva 4 neurony. Ostatní vrstvy, tedy obě GRU a plně propojená, mají 230 neuronů. Jedná se o hodnotu, která spolu se zvolenou délkou sekvence 1000 datových jednotek

v jedné sekvenci zajišťují dobrou konvergenci trénování, ale zároveň ne přílišnou, aby nedošlo k přetrénování. ReLu aktivační funkce v tomto případě pomohla zajistit stabilitu validace, jejíž hodnota byla bez aktivační funkce každý trénink víceméně náhodná a nepředvídatelná. Sigmoidní i tanh aktivační fuknce zde nemohly být použity kvůli již zmíněnému problému s explodujícími gradienty.

Jednotlivé hodnoty hyperparametrů byly zvoleny následovně:

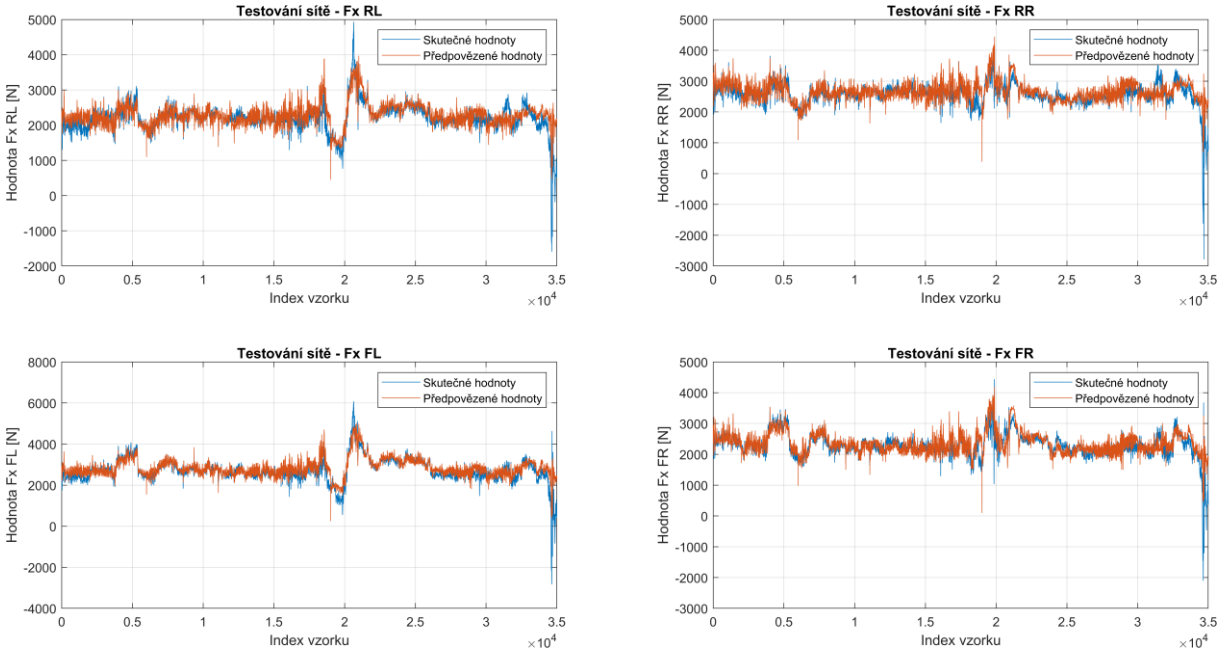
- Jako **optimalizátor** byl opět zvolen Adam, který zajišťuje ze všech řešičů nejlepší konvergenci sítě.
- **Počet epoch** byl zvolen jako 1000, kdy nad tuto hodnotu se síť začala přetrénovávat, a proto nemohla být zvolena hodnota vyšší.
- Hodnota **Gradient Threshold** je opět zvolena jako 1 pro kompromis mezi stabilitou a konvergencí sítě.
- Hyperparametr **Shuffle** byl nastaven na „every – epoch“ pro maximalizaci generalizace sítě.
- **Initial Learn Rate** byl nastaven na hodnotu 0,01. Jedná se o hodnotu vyšší rychlosti kvůli schopnosti sítě se učit, ale zároveň ne příliš vysoké, aby nedocházelo k přetrénování sítě. Při hodnotách vyšších od 0,04 a více již docházelo k explozi gradientů, což je dále umocněno větší sekvenci a vyšším počtem neuronů v jednotlivých vrstvách sítě.
- Hodnota **L2 Regularization** byla nastavena na 0,7 za účelem větší snahy opravit špatné váhy a snížení přetrénování.
- **Learn Rate Drop Period** byl zvolen na hodnotu 70, tedy snižování rychlosti učení jednou za 70 epoch s hodnotou **Learn Rate Drop Factor** 0,8 tedy se snižováním rychlosti učení o 20 %. Opět je to z toho důvodu, že je nutné pro zachování schopnosti trénování sítě ponechat vyšší rychlost učení, ale zároveň je potřeba snižovat rychlost učení pro snížení rizika exploze gradientů.
- Ostatní vedlejší hyperparametry zůstaly nastavené stejně jako u první sítě, tedy **Verbose** na hodnotě „1“, byla zadána validační data s frekvencí validace každých 50 epoch a hyperparametr **Execution Environment** nastaven pro zrychlení učení na GPU.

Na *obr. 45* je průběh trénování a validace vytvořené RNN. Modře je zobrazen průběh procesu trénování a černě validace. Jak již bylo zmíněno, ideálně by se měly hodnoty validace a tréninku na konci potkat, zde je ale viditelný rozdíl. Bohužel se nepodařilo v žádných ze zkoušených relací dosáhnout lepších hodnot validace.




*Obr. 45* Průběh trénování a validace vytvořené RNN pro predikci podélných sil na kolech vozidla

Průběhy predikovaných veličin na všech kolech vypadají podobně. Jedna z vad predikce je, že se zde nacházejí občasné odchylky, které ve skutečných datech nejsou. Dále si lze na *obr. 46* všimnout, že na konci predikce (index vzorku $3 \cdot 10^4$) se sice síť snaží predikovat jednotlivé odchylky, ovšem nedostatečně. Zbytek predikce průběhu veličin se však zdá být vcelku v pořádku i s absolutními hodnotami.




*Obr. 46* Predikce podélných sil na jednotlivých kolech pomocí vytvořené RNN sítě; RL – zadní levé kolo, RR – zadní pravé kolo, FL – přední levé kolo, FR – přední pravé kolo

# 12 OVĚŘENÍ ZÁVISLOSTI PŘESNOSTI SÍTĚ NA VYBRANÉ VSTUPNÍ VELIČINY

Opět je třeba zjistit závislost přesnosti trénování sítě na jednotlivých vstupních veličinách, aby se vědělo, na které veličiny je třeba dát důraz, nebo které jsou nedůležité a je možno je ze vstupních veličin odebrat, jako tomu bylo v předchozí vytvořené neuronové síti.

Znovu tedy bylo provedeno trénování sítě bez jednotlivých skupin vstupů. Postupně byly odebírány všechny skupiny, které jsou stejné jako u první vytvářené sítě, tedy:

- vstupy zrychlení ve všech osách (aX, aY, aZ),
- vstup úhlu sklonu vozidla (pitch angle),
- vstup úhlu náklonu vozidla (roll angle),
- vstupy úhlové rychlosti ve všech osách (wX, wY, wZ),
- vstupy rychlostí ve všech osách (vX, vY, vZ),
- vstupy zdvihu tlumičů na všech kolech (dmpFR, dmpFL, dmpRR, dmpRL).

Pro hodnocení bylo opět každé nastavení natrénováno pětkrát s metrikou RMSE jak trénovacích, tak validačních dat. Hodnoty z jednotlivých trénovacích relací jsou uvedeny v *tab. 6*.

*Tab. 6* Přehled výsledků trénovaných sítí pro predikce podélné síly na kolech s různým nastavením

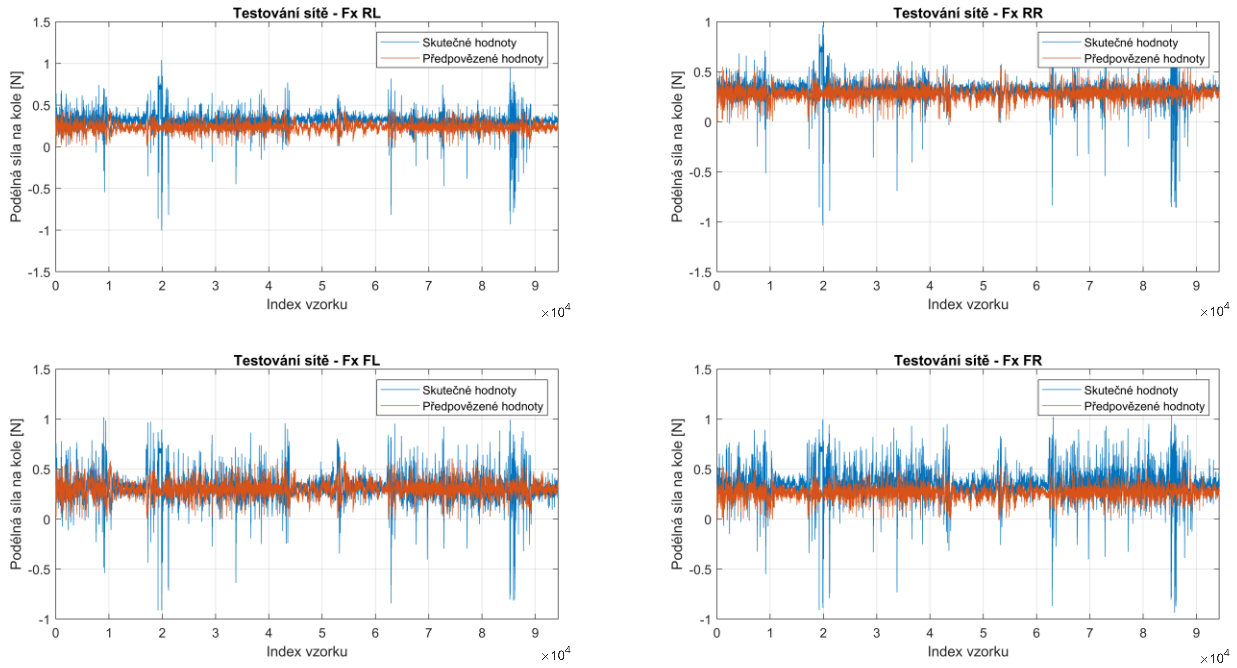
| Vstupní parametry | 1 | 1. validace | 2 | 2. validace | 3 | 3. validace | 4 | 4. validace | 5 | 5. validace | Test průměr | Validace průměr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| všechny | 263,74 | 2306,3 | 258,25 | 2290 | 264,99 | 2380,5 | 219,03 | 2213,4 | 267,92 | 2226,5 | 254,79 | 2283,34 |
| | 264,71 | | 277,31 | | 307,37 | | 270,98 | | 299,44 | | 283,96 | |
| | 317,50 | | 339,77 | | 347,41 | | 275,78 | | 340,02 | | 324,10 | |
| | 254,10 | | 292,90 | | 304,71 | | 262,86 | | 293,58 | | 281,63 | |
| všechny - aX, aY, aZ | 325,94 | 2481 | 341,91 | 2189,9 | 317,87 | 2303,8 | 317,07 | 2737,6 | 412,17 | 2458,5 | 342,99 | 2434,16 |
| | 361,58 | | 341,91 | | 342,27 | | 326,67 | | 447,63 | | 364,01 | |
| | 444,35 | | 473,42 | | 423,38 | | 412,28 | | 553,73 | | 461,43 | |
| | 398,89 | | 359,56 | | 360,78 | | 339,97 | | 464,24 | | 384,69 | |
| všechny - pitch | 281,87 | 2215,8 | 265,32 | 2238,9 | 307,50 | 2167,2 | 273,45 | 2209,7 | 274,80 | 2402,3 | 280,59 | 2246,78 |
| | 304,05 | | 294,84 | | 335,12 | | 313,98 | | 299,55 | | 309,51 | |
| | 337,14 | | 335,73 | | 368,50 | | 325,27 | | 356,53 | | 344,63 | |
| | 279,26 | | 291,28 | | 319,28 | | 297,06 | | 295,49 | | 296,48 | |
| všechny - roll | 257,47 | 2181,8 | 273,43 | 2173,4 | 267,76 | 2256,1 | 279,76 | 2099,5 | 236,20 | 2159,3 | 262,92 | 2174,02 |
| | 278,71 | | 308,66 | | 283,39 | | 309,08 | | 257,75 | | 287,52 | |
| | 347,40 | | 326,89 | | 354,77 | | 355,48 | | 314,67 | | 339,84 | |
| | 294,40 | | 292,41 | | 287,91 | | 307,41 | | 260,31 | | 288,49 | |
| všechny - wX, wY, wZ | 274,08 | 2123,3 | 247,94 | 2217,6 | 275,17 | 2331,5 | 300,61 | 2135,6 | 254,65 | 2242,9 | 270,49 | 2210,18 |
| | 297,71 | | 264,94 | | 343,88 | | 340,38 | | 269,44 | | 303,27 | |
| | 340,93 | | 299,14 | | 350,42 | | 349,09 | | 303,23 | | 328,56 | |
| | 274,55 | | 241,89 | | 321,14 | | 306,93 | | 252,67 | | 279,43 | |
| všechny - vX, vY, vZ | 231,14 | 2188,9 | 296,79 | 2156,7 | 261,70 | 2239,1 | 264,59 | 2247,6 | 269,02 | 2137,6 | 264,65 | 2193,98 |
| | 241,73 | | 333,17 | | 270,50 | | 265,66 | | 289,88 | | 280,19 | |
| | 302,71 | | 366,60 | | 311,73 | | 328,67 | | 328,83 | | 327,71 | |
| | 238,84 | | 325,86 | | 251,48 | | 254,60 | | 275,68 | | 269,29 | |
| všechny - dmp | 295,55 | 2398,7 | 320,39 | 2200,8 | 272,00 | 2528,1 | 283,08 | 2020,6 | 298,00 | 2299,6 | 293,81 | 2289,56 |
| | 335,26 | | 362,93 | | 341,32 | | 329,63 | | 366,09 | | 347,05 | |
| | 391,18 | | 407,16 | | 345,26 | | 375,07 | | 387,83 | | 381,30 | |
| | 350,50 | | 351,24 | | 323,93 | | 331,80 | | 366,37 | | 344,77 | |
| dmp + vZ + aZ + pitch + roll | 309,51 | 2180,4 | 299,65 | 2040,8 | 327,97 | 2220 | 362,07 | 2132,2 | 362,07 | 2132,2 | 332,25 | 2141,12 |
| | 322,62 | | 311,91 | | 350,47 | | 389,97 | | 389,97 | | 352,99 | |
| | 386,17 | | 392,73 | | 416,16 | | 469,96 | | 469,96 | | 426,99 | |
| | 322,54 | | 327,82 | | 345,38 | | 406,09 | | 406,09 | | 361,58 | |
| dmp + vZ + aX, aY, aZ + pitch + roll | 222,85 | 2189,8 | 234,35 | 2140,1 | 272,74 | 2258,8 | 222,85 | 2189,8 | 266,95 | 2146,5 | 243,95 | 2185 |
| | 241,33 | | 264,14 | | 298,93 | | 241,33 | | 294,08 | | 267,96 | |
| | 291,74 | | 313,15 | | 339,39 | | 291,74 | | 304,48 | | 308,10 | |
| | 254,89 | | 277,90 | | 284,68 | | 254,89 | | 263,32 | | 267,14 | |

Hodnota RMSE pro validaci je uvedena pouze jedna, protože se validuje síť, která je jedna pro všechny čtyři veličiny. Ovšem odhady podélných sil pro jednotlivá kola jsou hodnoceny pomocí RMSE zvlášť, a proto jsou pro každou relaci 4 hodnocení. V tomto případě nejsou výsledky tak jednoznačné jako tomu bylo u první trénované sítě. Nejhorší výsledky má síť bez vstupních veličin zrychlení ve všech směrech. Z toho lze konstatovat, že se jedná o stěžejní veličinu pro predikci. Zhoršení oproti základnímu nastavení se všemi vstupními veličinami přináší i odebrání všech vstupů zdvihu tlumičů. Ostatní odebrání veličin vedlo buď k mírnému zlepšení nebo stagnaci na stejném hodnocení jako základní nastavení. Následně byla intuitivně vzhledem k odhadované veličině sestavena ještě další 2 nastavení, první obsahuje vstupy z tlumičů, náklon vozidla, sklon vozidla a zrychlení a rychlost pouze v ose Z. Druhé natavení má pak navíc zrychlení ve všech osách.

Nejlepší výsledky pro validační část má s nejnižším RMSE nastavení, kde se nachází vstupy z tlumičů, sklon a náklon vozidla a rychlost a zrychlení pouze v ose Z. Ovšem v testovací části má toto nastavení jedno z nejhorších výsledků. Jako nejlepší nastavení bylo tedy vybráno nastavení poslední se vstupy z tlumičů, rychlostí v ose Z, zrychlením ve všech osách a sklonem a náklonem vozidla. Toto nastavení vykazuje nejlepší hodnocení v testovací oblasti a jedno z nejlepších hodnocení v oblasti validační. Pokud tedy vezmeme v úvahu obě části, má toto nastavení celkové výsledky nejlepší. Jako výsledná síť, která je výstupem této práce bude použita síť z 2. pokusu testování zmíněného nastavení viz *tab. 6* (zeleně vyznačeno) se stejnými hyperparametry a architekturou, které byly popsány v kapitole 11.

# 13 OTESTOVÁNÍ RNN PRO PREDIKCI PODÉLNÝCH SIL NA JEDNOTLIVÝCH KOLECH VOZIDLA

Stejně jako u první sítě je potřeba vytvořenou RNN pro predikci podélných sil otestovat na jiném datasetu, než na kterém byla vytvářena. Pro otestování byl použit stejný dataset jako v prvním případě, tedy dataset obsahující 943000 časových kroků. Pro otestování musí být dataset upraven stejně jako vstupní hodnoty do tréninku neuronové sítě. Musí být tedy odfiltrovány pomocí filtru s dolní propustí, rozděleny do sekvencí po 1000 hodnotách a následně normalizovány. Výsledek testování je na *obr. 47*.




*Obr. 47* Testování vytvořené RNN pro predikci podélných sil na kolech vozidla na jiném datasetu

Testování na jiném datasetu ukazuje, že síť dokáže vcelku dobře predikovat základní průběh dat, ovšem absolutní hodnoty jsou úplně správné pouze u jednoho ze 4 kol. U zbylých 3 kol je průběh mírně posunut o konstantní hodnotu, pro každé kolo o jinak velkou. Celkový průběh vypadá v pořádku, ale pokud se zaměříme na detaily, najdou se zde chyby. Síť odhaduje základní průběh, ale celkem často zde chybí náhlé odchylky. Největším na první pohled viditelným problémem jsou zde však již zmíněné nesprávné predikované absolutní hodnoty u 3 ze 4 kol.

Tento problém byl intenzivně řešen. Bylo vyzkoušeno odstranění normalizace za účelem zachování hodnoty „výškového“ posunu dat vstupních hodnot jako tomu bylo u předchozí sítě, ovšem tentokrát již tato úprava nezabrala. Síť se totiž nedokázala z dat bez normalizace vůbec učit. Dále tedy bylo vyzkoušeno vytvoření vlastní sítě pro každé kolo zvlášť, výsledky však byly stejné – opět byly správné absolutní hodnoty pouze u předního levého kola a predikce u ostatních kol byly posunuté. Nakonec bylo vyzkoušeno i prohození datasetů, kdy na testovacím datasetu byla síť natrénována a na trénovacím datasetu otestována, ale opět to nevedlo k žádné

změně. Jelikož žádný z těchto postupů nevedl ke zlepšení predikce absolutních hodnot jedná se nejspíš o rozdílné hodnoty v použitých datasetech na jednotlivých kolech. Řešením by mohlo být spojení obou datasetů a učení sítě na obou datasetech najednou. Tím by se odhadované hodnoty mohly zprůměrovat mezi oběma datasety a minimalizovala by se tak hodnota posunutí. Jako úplné řešení problému by se to však považovat nedalo.

Konečné výsledky natrénované RNN pro predikci podélných sil na kolech jsou i přes mírné posunutí u některých kol přijatelné. I když to vizuálně na první pohled nemusí vypadat tak dobře, jako tomu je u první natrénované sítě pro predikci směrové úchylky, u této sítě se jedná o řádově vyšší odhadované hodnoty, a proto jsou chyby v absolutních hodnotách vyšší. I když po přepočtu na procentuální chyby vychází obě sítě podobně. Výsledné hodnoty RMSE pro jednotlivá kola jsou:

- 1276 pro zadní levé kolo,
- 943 pro zadní pravé kolo,
- 1482 pro přední levé kolo,
- 1492 pro přední pravé kolo.

Jak je zmíněno výše, hodnoty jsou řádově vyšší, než byly hodnoty RMSE u sítě pro predikci směrové úchylky, což je ovšem způsobeno právě řádově rozdílnými hodnotami odhadovaných veličin. Zajímavé také je, že pravé zadní kolo má lepší RMSE než levé přední kolo, které je vizuálně lepší. Zde se opět promítá náchylnost hodnocení RMSE na největší odchylky, kterých je u predikce hodnot podélné síly předního levého kola více než u pravého zadního kola. Průměrná hodnota RMSE ze všech kol je 1298,25 což je v porovnání s průměrnou hodnotou podélných sil na všech kolech 26467,5 N dobrá hodnota, přestože to na první pohled z grafů testování nevypadá.

# ZÁVĚR

Tato práce se zabývá vytvořením neuronových sítí s cílem ověřit využití sítí pro odhad dynamických veličin automobilu. Původním plánem bylo natrénování jedné neuronové sítě pro odhad pěti dynamických veličin, ovšem kvůli zachování přesnosti predikce sítě a odlišnosti predikovaných veličin byla zvolena varianta natrénování dvou neuronových sítí.

Vzhledem k povaze dynamických veličin byla pro natrénování obou sítí zvolena architektura rekurentní neuronové sítě, která umožňuje práci se sekvenčními daty. První vytvořená síť byla natrénována pro predikci jedné veličiny – úhlu směrové úchylky, zatímco druhá vytvořená síť byla natrénována pro predikci čtyř veličin – podélných sil na všech kolech automobilu.

Důležitým krokem při tvorbě neuronových sítí je zpracování vstupních dat, díky kterému jsou z naměřených dat odstraněny nežádoucí informace pro zjednodušení trénování neuronové sítě. Během tohoto procesu bylo v práci zjištěno, že normalizace dat, jakožto často používaná úprava vstupních veličin, může mít i negativní vliv, a je proto dobré zvážit její použití. Při tvorbě neuronových sítí v praktické části vyšlo, že pro řešený problém je dobré využít normalizaci při absolutních hodnotách dynamických veličin v desítkách a výš. Při hodnotách nižších totiž normalizace způsobovala, že data ztratila jeden z rysů, které v sobě mají.

Dalším krokem je samotné sestavení architektury neuronové sítě a volba vhodných hyperparametrů a jejich hodnot. Vhodné hodnoty se liší na základě řešeného problému, ale vzhledem k tomu, že neexistují žádné matematické vztahy k určení správných hodnot hyperparametrů, musí být voleny empiricky. To ovšem ztěžuje volbu počátečních hodnot, obzvlášť když se jedná o začátky řešení daného problému.

Iteračním způsobem tedy byly voleny hodnoty hyperparametrů i architektura neuronových sítí, dokud nebylo dosaženo uspokojivých výsledků. Během tohoto iteračního postupu byly zjištěny důležité poznatky vztahující se k tvorbě neuronových sítí pro odhad dynamických veličin.

Jelikož síť pracuje se sekvenčními daty, je potřeba zvolit délku sekvence, po které je bude síť zpracovávat. Jedná se o hodnotu, na které je závislá kvalita sítě. Obecně je dobré zvolit délku dle sledovaného dynamického jevu tak, aby byl schopen se v dané sekvenci dostatečně projevit. Dále je vhodné zamíchat sekvence mezi sebou, což umožní zachovat v datech sledované dynamické jevy, ale zároveň pomoct k lepší generalizaci sítě.

Co se týče vstupních veličin, tak bylo na začátku zvoleno více základních vstupních veličin s očekáváním lepších výsledků při vyšším počtu vstupů. To se ovšem ukázalo jako mylný předpoklad. Pokud jsou ve vstupech obsaženy veličiny, ze kterých se síť dokáže naučit rysy výstupních dat, pak další vstupy spíše zhoršují kvalitu predikce. V této práci byly vybrány jako vstupy základní dynamické veličiny měřitelné základními metodami za účelem zlepšení dostupnosti určení těžko měřitelných dynamických veličin automobilu. Následně byla pomocí několika trénovacích relací zjištěna závislost kvality predikce sítě na vstupech a díky tomu mohly být nadbytečné vstupy odstraněny. U vytvořené rekurentní neuronové sítě pro odhad úhlu směrové úchylky zůstaly pouze tři vstupy, konkrétně rychlost ve všech třech osách kartézského souřadného systému. U druhé vytvořené sítě pro odhad podélných sil na všech kolech automobilu zůstalo již vstupních veličin více, přičemž největší závislost vykazuje síť na vstupech zrychlení ve všech osách kartézského souřadného systému. Obecně však nelze říct, že by nějaká vstupní veličina byla nepoužitelná, protože každá z výstupních veličin je závislá na vstupních veličinách jinak. Použitelnost vstupu tedy záleží na odhadované dynamické veličině.

Výsledky první vytvořené neuronové sítě pro predikci úhlu směrové úchylky jsou velmi dobré, kdy až na pár výjimek, kdy síť nedokáže predikovat rychlé vysoké vrcholky, síť správně predikuje jak průběh, tak absolutní hodnotu. Funkčnost sítě byla otestována jak na testovacích datech, tak i na jiném datasetu, než na kterém byla síť trénována. RMSE testované sítě je 0,08 což je vůči průměrné hodnotě úhlu směrové úchylky 0,43 ° přijatelná hodnota s tím, že jsou v průběhu již zmíněné špatně predikované rychlé vysoké vrcholky, které mají dopad na konečnou hodnotu RMSE. Pro porovnání hodnota MAE, která není tolik náchylná na odchylky, je 0,0099, což je po přepočtu chyba cca 2 %.

Výsledky druhé vytvořené neuronové sítě pro predikci podélných sil na všech kolech automobilu jsou sice číselně srovnatelné, ale opticky již ne tak kvalitní. Síť byla opět ověřena na testovacích datech i dalším datasetu. Celkově bylo síť těžší naladit a musela mít složitější architekturu, což souvisí s počtem výstupních veličin. Přestože se jedná o stejnou veličinu na každém kole, jsou na každém kole tyto hodnoty rozdílné, a to ztěžuje návrh sítě. Síť dokáže predikovat vcelku dobře průběh jednotlivých sil na každém kole, ovšem absolutní hodnoty jsou úplně správné pouze pro jedno z kol. Ostatní kola jsou posunutá o konstantu, přičemž každé z kol o jinou. Ještě více je zde síť náchylná na rychlé vysoké odchylky. Průměrná hodnota RMSE ze všech kole je 1298 a MAE 636, což je v porovnání s průměrnou hodnotou podélných sil 26468 N chyba 2,4 %.

Přestože výsledky predikcí nejsou špatné, stále je co vylepšovat. Obě sítě jsou trénovány pouze na jedné cestě automobilu. Pokud by však trénovací dataset byl zvětšen o další cesty, byla by síť více generalizována a mohla by být preciznější. Další věcí je, že jsou obě sítě natrénovány pouze na jednom typu automobilu a není tedy možné je využít obecně pro jakékoliv vozidlo. Tento problém by šel vyřešit natrénováním sítí na datech z různých automobilů. Oba případy zvětšení datasetů však velmi zvýší časovou náročnost trénování obou sítí. Vzhledem k nestálosti výsledků druhé neuronové sítě u jednotlivých kol by bylo dobré zkusit vytvořit síť ještě hlubší a celkově ji zesložitit. Avšak kvůli tomu, že čím složitější síť je, tím více časově náročné je postupně iterovat ke správné architektuře a hodnotám hypeparametrů sítě, není již tato úprava pro sjednocení predikcí na jednotlivých kolech předmětem této práce.

Celkově však dobré výsledky práce ukazují, že lze neuronové sítě využít pro predikci dynamických veličin automobilu. Pokud bude k natrénování sítě dostupné dostatečně velké množství dat, bude síť i dobře generalizována a bude mít správné predikce. Predikce sítě těžce měřitelných veličin díky vstupům z dostupně měřitelných vstupů by po správném vyladění sítě pro predikci konkrétní dynamické veličiny mohly nahradit drahá měření.

# POUŽITÉ INFORMAČNÍ ZDROJE

[1] Základní automobilové názvosloví. Online. *ROZMĚRY VOZIDEL*. 1979. Dostupné z: https://nahledy.normy.biz/n.php?i=6491. [cit. 2024-05-21].

[2] ISO 612:1978: Road vehicles — Dimensions of motor vehicles and towed vehicles — Terms and definitions. Online. In: . Edition 1, 1978. Dostupné z: https://www.iso.org/standard/4729.html. [cit. 2024-05-21].

[3] *Základní rozměry vozidel*. Online. Autolexicon.net. C2023. Dostupné z: https://www.autolexicon.net/cs/articles/zakladni-rozmery-vozidel/. [cit. 2023-06-10].

[4] VLK, František. *Dynamika motorových vozidel.* 2. vyd. Brno: František Vlk, 2003. ISBN 80-239-0024-2.

[5] PORTEŠ, Petr. Zatížení náprav. Online. In: . 2023. Dostupné z: https://moodle.vut.cz/mod/folder/view.php?id=238726. [cit. 2024-05-21].

[6] MATHWORKS. *What Is a Neural Network?* Online. Https://www.mathworks.com/. C2023. Dostupné z: https://www.mathworks.com/discovery/neural-network.html. [cit. 2024-05-21].

[7] AK, Abel. Regression vs. Classification: Understanding the Differences and Use Cases. Online. *Medium*. Roč. 2023. Dostupné z: https://medium.com/@abelkuriakose/regression-vs-classification-understanding-the-differences-and-use-cases-6cd51c236112. [cit. 2024-01-10].

[8] *Introduction to Convolutional Neural Networks (CNN).* Online. Analytics Vidhya. 2021. Dostupné z: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/. [cit. 2023-09-30].

[9] N JOSHI, Srivatsa; H KUMAR, Vishwas; VENKATARAMANAN, Vishaka a C S, Kaliprasad. A Review on Neural Networks and its Applications. Online. 2023, s. 59-70. ISSN 2229-6964. Dostupné z: https://www.researchgate.net/publication/374155705_A_Review_on_Neural_Networks_and_its_Applications. [cit. 2023-09-30].

[10] NING, Hongwei; LIU, Sheng; ZHU, Qifei a ZHOU, Teng. Convolutional neural network in rice disease recognition: accuracy, speed and lightweight. Online. *Frontiers in Plant Science*. 2023, roč. 14. ISSN 1664-462X. Dostupné z: https://doi.org/10.3389/fpls.2023.1269371. [cit. 2023-11-17].

[11] JIANG, Feng; LU, Yang; CHEN, Yu; CAI, Di a LI, Gongfa. Image recognition of four rice leaf diseases based on deep learning and support vector machine: a review. Online. *Computers and Electronics in Agriculture*. 2020, roč. 179, č. 14, s. 21415-21481. ISSN 01681699. Dostupné z: https://doi.org/10.1016/j.compag.2020.105824. [cit. 2023-11-17].

[12] YEOM, Seul-Ki; SEEGERER, Philipp; LAPUSCHKIN, Sebastian; BINDER, Alexander; WIEDEMANN, Simon et al. Pruning by explaining: A novel criterion for deep neural network pruning. Online. *Pattern Recognition*. 2021, roč. 115. ISSN 00313203. Dostupné z: https://doi.org/10.1016/j.patcog.2021.107899. [cit. 2023-11-18].

[13] ZHANG, Rongzhao a CHUNG, Albert C.S. MedQ: Lossless ultra-low-bit neural network quantization for medical image segmentation. Online. *Medical Image Analysis*. 2021, roč. 73. ISSN 13618415. Dostupné z: https://doi.org/10.1016/j.media.2021.102200. [cit. 2023-11-18].

[14] ANVARJON, Tursunov; MUSTAQEEM a KWON, Soonil. Deep-Net: A Lightweight CNN-Based Speech Emotion Recognition System Using Deep Frequency Features. Online. *Sensors*. 2020, roč. 20, č. 18. ISSN 1424-8220. Dostupné z: https://doi.org/10.3390/s20185212. [cit. 2023-11-18].

[15] S., Gayathri; GOPI, Varun P. a PALANISAMY, P. A lightweight CNN for Diabetic Retinopathy classification from fundus images. Online. *Biomedical Signal Processing and Control*. 2020, roč. 62. ISSN 17468094. Dostupné z: https://doi.org/10.1016/j.bspc.2020.102115. [cit. 2023-11-19].

[16] AWATI, Rahul. *Convolutional neural network (CNN)*. Online. TechTarget. 2023. Dostupné z: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network. [cit. 2023-11-19].

[17] FERNÁNDEZ, Juan; CHIACHÍO, Juan; BARROS, José; CHIACHÍO, Manuel a KULKARNI, Chetan S. Physics-guided recurrent neural network trained with approximate Bayesian computation: A case study on structural response prognostics. Online. 2024, roč. 243. ISSN 09518320. Dostupné z: https://doi.org/10.1016/j.ress.2023.109822. [cit. 2024-05-21].

[18] PASCANU, Razvan; MIKOLOV, Tomas a BENGIO, Yoshua. On the difficulty of training Recurrent Neural Networks. Online. 2013. Dostupné z: https://doi.org/10.48550/arXiv.1211.5063. [cit. 2024-05-21].

[19] CHATURVEDI, Naina. Recurrent Neural Network with Keras. Online. *DataDrivenInvestor*. 2021. Dostupné z: https://medium.datadriveninvestor.com/recurrent-neural-network-with-keras-b5b5f6fe5187. [cit. 2023-10-03].

[20] GOODFELLOW, Ian; BENGIO, Yoshua a COURVILLE, Aaron. *Deep learning*. Adaptive computation and machine learning (MIT Press). Cambridge: MIT Press, [2016]. ISBN 978-026-2035-613.

[21] C. LIPTON, Zachary; BERKOWITZ, John a ELKAN, Charles. A Critical Review of Recurrent Neural Networks for Sequence Learning. Online. 2015. Dostupné z: https://arxiv.org/abs/1506.00019. [cit. 2024-01-10].

[22] GREFF, Klaus; K. SRIVASTAVA, Rupesh; KOUTNÍK, Jan; R. STEUNEBRINK, Bas a SCHMIDHUBER, Jürgen. LSTM: A Search Space Odyssey. Online. Roč. 2017. Dostupné z: https://doi.org/10.48550/arXiv.1503.04069. [cit. 2024-01-10].

[23] QAMAR, Roheen a ALI ZARDARI, Baqar. Artificial Neural Networks: An Overview. Online. *Mesopotamian Journal of Computer Science*. S. 130-139. Dostupné z: https://doi.org/10.58496/MJCSC/2023/015. [cit. 2023-10-03].

[24] NEWELL, Allen. A Step toward the Understanding of Information Processes: Perceptrons . An Introduction to Computational Geometry. Marvin Minsky and Seymour Papert. M.I.T. Press, Cambridge, Mass., 1969. vi 258 pp., illus. Cloth, $12; paper, $4.95. Online. *Science*. 1969, roč. 165, č. 3895, s. 780-782. ISSN 0036-8075. Dostupné z: https://doi.org/10.1126/science.165.3895.780. [cit. 2024-01-11].

[25] HILTEMANN, Saskia; RASCHE, Helena; GLADMAN, Simon; HOTZ, Hans-Rudolf; LARIVIÈRE, Delphine et al. Galaxy Training: A powerful framework for teaching!. Online. *PLOS Computational Biology*. 2023, roč. 19, č. 1. ISSN 1553-7358. Dostupné z: https://doi.org/10.1371/journal.pcbi.1010752. [cit. 2024-01-11].

[26] KYAW MYINT, Aung. In depth explanation of FeedForward in Neural Network mathematically. Online. *Medium*. 2019. Dostupné z: https://medium.com/analytics-vidhya/in-depth-explanation-of-feedforward-in-neural-network-mathematically-448092216b63. [cit. 2024-01-15].

[27] , Turing. *What Is the Necessity of Bias in Neural Networks?* Online. TURING. C2024. Dostupné z: https://www.turing.com/kb/necessity-of-bias-in-neural-networks. [cit. 2024-01-15].

[28] MELZI, S. a SABBIONI, E. On the vehicle sideslip angle estimation through neural networks: Numerical and experimental results. Online. *Mechanical Systems and Signal Processing*. 2011, roč. 25, č. 6, s. 2005-2019. ISSN 08883270. Dostupné z: https://doi.org/10.1016/j.ymssp.2010.10.015. [cit. 2023-11-07].

[29] CHINDAMO, D. a GADOLA, M. Estimation of vehicle side-slip angle using an artifical neural network. Online. 2018, č. 166. Dostupné z: https://www.matec-conferences.org/articles/matecconf/pdf/2018/25/matecconf_icmaa2018_02001.pdf. [cit. 2023-11-07].

[30] ZHOU, Zhisong; WANG, Yafei; JI, Qinghui; WELLMANN, Daniel; ZENG, Yifan et al. A Hybrid Lateral Dynamics Model Combining Data-driven and Physical Models for Vehicle Control Applications. Online. *IFAC-PapersOnLine*. 2021, roč. 54, č. 20, s. 617-623. ISSN 24058963. Dostupné z: https://doi.org/10.1016/j.ifacol.2021.11.240. [cit. 2023-11-11].

[31] BAHETI, Pragati. A Simple Guide to Data Preprocessing in Machine Learning. Online. *V7*. 2021. Dostupné z: https://www.v7labs.com/blog/data-preprocessing-guide. [cit. 2024-01-17].

[32] WALLENTOWITZ, Henning Wallentowitz; KÖHN, Philip a HOLDMANN, Peter. Dynamic Properties of Tyres - Testing and Simulation. Online. *SAE Transactions*. 1999, roč. 108, č. SECTION 6, article Vol. 108, s. 1548-1553. Dostupné z: https://www.jstor.org/stable/44668030. [cit. 2024-01-20].

[33] LUGNER, P. a PLÖCHL, M. Modelling in vehicle dynamics of automobiles. Online. ZAMM - *Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*. 2004, roč. 84, č. 4, s. 219-236. ISSN 0044-2267. Dostupné z: https://doi.org/10.1002/zamm.200310108. [cit. 2024-01-20].

[34] *Lowpass*. Online. MathWorks. C2024. Dostupné z: https://www.mathworks.com/help/signal/ref/lowpass.html#d120e85766. [cit. 2024-01-21].

[35] BAHETI, Pragati. Train Test Validation Split: How To & Best Practices. Online. *V7*. 2021. Dostupné z: https://www.v7labs.com/blog/train-validation-test-set. [cit. 2024-01-23].

[36] *What Is a Recurrent Neural Network?* Online. MathWorks. C2024. Dostupné z: https://www.mathworks.com/discovery/rnn.html. [cit. 2024-02-05].

[37] *TrainingOptions*. Online. MathWorks. C2024. Dostupné z: https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu59f0q_sep_mw_6dcd8e56-4de5-497d-a48e-0f6f804f57f6_head. [cit. 2024-03-05].

[38] HERMANSDORFER, Leonhard; TRAUTH, Rainer; BETZ, Johannes a LIENKAMP, Markus. End-to-End Neural Network for Vehicle Dynamics Modeling. Online. 2020 6th IEEE *Congress on Information Science and Technology (CiSt)*. 2020, s. 407-412. ISBN 978-1-7281-6646-9. Dostupné z: https://doi.org/10.1109/CiSt49399.2021.9357196. [cit. 2024-03-09].

# SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

| | | |
|---|---|---|
| $\mu_V$ | [-] | Součinitel valivé přilnavosti |
| $\mu_\sigma$ | [-] | Součinitel skluzové přilnavosti |
| *1D* | [-] | Jednorozměrný prostor |
| *2D* | [-] | Dvojrozměrný prostor |
| *Adam* | [-] | Adaptivní odhad momentu |
| $a_x$ | [m·s$^{-2}$] | Podélné zrychlení vozidla |
| *aX* | [m·s$^{-2}$] | Zrychlení ve směru osy x |
| *aY* | [m·s$^{-2}$] | Zrychlení ve směru osy y |
| *aZ* | [m·s$^{-2}$] | Zrychlení ve směru osy z |
| *b* | [m] | Vzdálenost přední nápravy od těžiště |
| *b* | [-] | Zkreslení |
| *c* | [m] | Vzdálenost zadní nápravy od těžiště |
| $C_{M\alpha K}$ | [N·m rad$^{-1}$] | Vratná tuhost pneumatiky |
| *CNN* | [-] | Konvoluční neuronová síť |
| *CPU* | [-] | Centrální procesorová jednotka |
| $c_x$ | [-] | Součinitel vzdušného odporu |
| $C_{\alpha K}$ | [N·rad$^{-1}$] | Směrová tuhost pneumatiky |
| $C_\xi$ | [N/mm] | Klopná tuhost pneumatiky |
| $D_A$ | [N] | Aerodynamický odpor |
| $d_h$ | [m] | Vzdálenost působiště podélné síly působící v tažném zařízení |
| *dmpFL* | [mm] | Zdvih tlumiče levého předního kola |
| *dmpFR* | [mm] | Zdvih tlumiče pravého předního kola |
| *dmpRL* | [mm] | Zdvih tlumiče levého zadního kola |
| *dmpRR* | [mm] | Zdvih tlumiče pravého zadního kola |
| *e* | [m] | Svislá vzdálenost kola předsunutí radiální reakce |
| *f* | [-] | Aktivační funkce |
| *FFT* | [-] | Rychlá Fourierova transformace |
| $f_K$ | [-] | Součinitel valivého odporu |
| $F_K$ | [N] | Hnací síla |
| $F_{Kmax}$ | [N] | Maximální hnací síla |
| *FNN* | [-] | Dopředná neuronová síť |
| $f_{Nyquist}$ | [Hz] | Nyquistova frekvence |

| | | |
|---|---|---|
| $f_s$ | [Hz] | Vzorkovací frekvence |
| *FxFL* | [N] | Podélná síla působící na levém předním kole |
| *FxFR* | [N] | Podélná síla působící na pravém předním kole |
| *FxRL* | [N] | Podélná síla působící na levém zadním kole |
| *FxRR* | [N] | Podélní síla působící na pravém zadním kole |
| *g* | [m·s$^{-2}$] | Tíhové zrychlení |
| *G* | [N] | Tíha vozidla |
| *GPU* | [-] | Grafický procesor |
| *GRU* | [-] | Gated Recurrent Unit (Uzavřená opakující se jednotka) |
| *h* | [m] | Výška těžiště |
| *h* | [-] | Finální vektor hodnot skryté vrstvy |
| $h_0$‘ | [m] | Vzdálenost určující polohu těžiště vůči přední části vozidla |
| $h_0$‘ | [m] | Vzdálenost určující polohu těžiště vůči zadní části vozidla |
| $h_1$ | [-] | Finální složka vektoru skryté vrstvy |
| $h_1$‘ | [-] | Složka vektoru skryté vrstvy |
| $h_2$ | [-] | Finální složka vektoru skryté vrstvy |
| $h_2$‘ | [-] | Složka vektoru skryté vrstvy |
| $h_3$ | [-] | Finální složka vektoru skryté vrstvy |
| $h_3$‘ | [-] | Složka vektoru skryté vrstvy |
| $h_a$ | [m] | Výška působiště aerodynamického odporu |
| $h_h$ | [m] | Výška působiště podélné síly působící v tažném zařízení |
| $H_{Kmax}$ | [N] | Maximální přenositelná obvodová síla mezi kolem a vozovkou |
| $H_{K\sigma}$ | [N] | Obvodová síla |
| $h_P$‘‘ | [m] | Vzdálenost určující polohu těžiště vůči přední části vozidla |
| $h_t$ | [-] | Průběžný aktivační vektor |
| $h_{t-1}$ | [-] | Předchozí skrytý stav |
| $h_Z$‘‘ | [m] | Vzdálenost určující polohu těžiště vůči zadní části vozidla |
| $i_C$ | [-] | Celkový převod mezi motorem a hnacími koly |
| $i_r$ | [-] | Převod rozvodovky |
| $J_{Ki}$ | [kg·m$^2$] | Moment setrvačnosti hnacích kol |
| $J_m$ | [kg·m$^2$] | Moment setrvačnosti rotujících částí |
| $J_p$ | [kg·m$^2$] | Moment setrvačnosti převodovky |
| *k* | [-] | Délka nebo šířka konvolučního jádra |

| | | |
|---|---|---|
| $L$ | [m] | Rozvor vozidla |
| $l$ | [m] | Rozvor automobilu |
| $L_{Af}$ | [N] | Aerodynamický vztlak působící na přední nápravu |
| $L_{Ar}$ | [N] | Aerodynamický vztlak působící na zadní nápravu |
| *L-BFGS* | [-] | Omezená paměť – Broyden – Fletcher – Goldfarb – Shanno |
| $l_P$ | [m] | Vzdálenost předních kol od těžiště automobilu |
| *LSTM* | [-] | Long Short Term Memory (Dlouhodobě krátkodobá paměť) |
| $l_Z$ | [m] | Vzdálenost zadních kol od těžiště automobilu |
| $m$ | [kg] | Hmotnost vozidla |
| $m‘$ | [kg] | Hmotnost odpružené části vozidla |
| $m‘‘_P$ | [kg] | Hmotnost přední nápravy vozidla |
| *MAE* | [-] | Střední absolutní chyba |
| $M_K$ | [N·m] | Moment na hnacích kolech |
| *MSE* | [-] | Střední kvadratická chyba |
| $M_{SK}$ | [N·m] | Vratný moment pneumatiky |
| $n_s$ | [m] | Závlek pneumatiky |
| $O_{fK}$ | [N] | Valivý odpor kola |
| $O_S$ | [N] | Odpor stoupání |
| $O_Z$ | [N] | Odpor zrychlení |
| $P$ | [-] | Sdružovací funkce |
| $P_K$ | [W] | Výkon vozidla potřebný pro překonání odporů |
| $p_P$ | [m] | Vzdálenost určující polohu těžiště vůči přední části vozidla |
| $p_Z$ | [m] | Vzdálenost určující polohu těžiště vůči zadní části vozidla |
| $R$ | [-] | Aktivační funkce |
| $r_d$ | [m] | Dynamický poloměr kola |
| *ReLU* | [-] | Usměrňovací aktivační funkce |
| $R_{hx}$ | [N] | Podélná síla působící v tažném zařízení |
| $R_{hz}$ | [N] | Vertikální síla působící v tažném zařízení |
| *RMSE* | [-] | Odmocnina ze střední kvadratické chyby |
| *RMSProp* | [-] | Střední kvadratické šíření |
| *RNN* | [-] | Rekurentní neuronová síť |
| $r_t$ | [-] | Resetovací brána |
| *SGDM* | [-] | Stochastický gradientní sestup s hybností |

| | | |
|---|---|---|
| $S_K$ | [N] | Boční vodící síla kola |
| $S_x$ | [$m^2$] | Čelní plocha vozidla |
| *tanh* | [-] | Funkce hyperobolického tangensu |
| $t_P$ | [m] | Vzdálenost určující polohu těžiště vůči přední části vozidla |
| $t_Z$ | [m] | Vzdálenost určující polohu těžiště vůči zadní části vozidla |
| $v$ | [$m \cdot s^{-1}$] | Rychlost vozidla |
| $v_K$ | [$m \cdot s^{-1}$] | Rychlost kola vozidla |
| $v_r$ | [$m \cdot s^{-1}$] | Náporová rychlost proudění vzduchu |
| $vX$ | [$m \cdot s^{-1}$] | Rychlost ve směru osy x |
| $vY$ | [$m \cdot s^{-1}$] | Rychlost ve směru osy y |
| $vZ$ | [$m \cdot s^{-1}$] | Rychlost ve směru osy z |
| $W$ | [N] | Tíha vozidla |
| $w$ | [-] | Váhový vektor |
| $W_f$ | [N] | Zatížení přední nápravy |
| $W_h$ | [-] | Váhová matice průběžného aktivačního vektoru |
| $W_{im}$ | [-] | I – tá složka spojující matice |
| $W_n$ | [-] | Složky váhová matice |
| $W_r$ | [N] | Zatížení zadní nápravy |
| $W_r$ | [-] | Váhová matice resetovací brány |
| $wX$ | [$rad \cdot s^{-1}$] | Úhlová rychlost kolem osy x |
| $wY$ | [$rad \cdot s^{-1}$] | Úhlová rychlost kolem osy y |
| $W_z$ | [-] | Váhová matice brány aktualizace |
| $wZ$ | [$rad \cdot s^{-1}$] | Úhlová rychlost kolem osy z |
| $x$ | [-] | Pixel obrazu |
| $x_K$ | [-] | Podélná osa vozidla |
| $x_n$ | [-] | Složka vektoru vstupů |
| $x_t$ | [-] | Aktuální vstup |
| $Y_K$ | [N] | Boční síla |
| $Z_K$ | [N] | Radiální reakce vozovky |
| $Z_{Pstat}$ | [N] | Svislé zatížení předního kola při ustálené poloze |
| $z_t$ | [-] | Brána aktualizace |
| $Z_{Zstat}$ | [N] | Svislé zatížení zadního kola při ustálené poloze |
| $\alpha$ | [°] | Úhel směrové úchylky |

| | | |
|---|---|---|
| $\alpha_K$ | [°] | Úhel směrové úchylky |
| $\beta$ | [-] | Váhový koeficient |
| $\Delta Z_P$ | [N] | Změna svislého zatížení předního kola |
| $\Delta Z_Z$ | [N] | Změna svislého zatížení zadního kola |
| $\eta$ | [-] | Celková mechanická účinnost převodovky a rozvodovky |
| $\xi$ | [°] | Úhel naklopení kola |
| $\rho$ | [kg·m$^3$] | Měrná hmotnost vzduchu |
| $\Phi$ | [-] | Aktivační funkce |

# SEZNAM PŘÍLOH