RĪGAS TEHNISKĀ UNIVERSITĀTE

Mašīnzinību, transporta un aeronautikas fakultāte

Mehānikas un mašīnbūves institūts

Teorētiskās mehānikas un materiālu pretestības katedra

Reinis Freibergs

Akadēmiskā bakalaura studiju programma

„Inženiertehnika, mehānika un mašīnbūve”

(stud. apl. nr. 191RMB002)

Salīdzinošā analīze dubultā svārsta sistēmas modelēšanai izmantojot dziļo neironu tīklu metodes.

Bakalaura darbs

Zinātniskais vadītājs:

Dr.sc.ing., pētnieks

Ē. URTĀNS

Rīga - 2022

DARBA IZPILDES UN NOVĒRTĒJUMA LAPA

Bakalaura darbs izstrādāts ***Teorētiskās mehānikas un materiālu pretestības katedrā***. Ar parakstu apliecinu, ka visi izmantotie materiāli ir norādīti literatūras sarakstā un iesniegtais darbs ir oriģināls.

Darba autors:

stud. **R. Freibergs** . . . . . .
(paraksts, datums)

Bakalaura darbs ieteikts aizstāvēšanai:

Zinātniskais vadītājs:

Dr.sc.ing., pētnieks **Ē. Urtāns** . . . . . .
(paraksts, datums)

Bakalaura darbs pielaists aizstāvēšanai:

Teorētiskās mehānikas un materiālu pretestības katedras vadītājs:

Dr.sc.ing., prof. **A. Krasņikovs** . . . . . .
(paraksts, datums)

Bakalaura darbs aizstāvēts Gala pārbaudījumu komisijas . . . . . gada . . . . . sēdē un novērtēts ar atzīmi (. . . . .) . . . . . .

Mehānikas un mašīnbūves institūta Gala pārbaudījumu komisijas

sekretāre . . . . . .
(paraksts, datums)

ANOTĀCIJA

Statistika liecina, ka pēdējo piecu gadu laikā augusi popularitāte gan dziļajos mākslīgo neironu tīklos balstītām metodēm kopumā, gan arī to pielietojumiem inženierzinātnēs.

Darbā iezīmētas dziļo mākslīgo neironu tīklu īpašības un ar tiem praktiski modelēta dubultā svārsta sistēma, par pamatu izmantojot datu kopu, kas ģenerēta no video ierakstiem. Rezultātu salīdzinājumam līdzīga modelēšana veikta ar sistēmu aprakstošiem diferenciālvienādojumiem. Salīdzinātas abu metožu spējas prognozēt nākamās 2s no dubultā svārsta kustības, informācijai izmantojot datus par iepriekšējās sekundes kustību, kā arī novērtētas modeļu spējas iekļauties noteiktā prognožu kļūdas intervālā.

Rezultāti liecina, ka labākais dziļo mākslīgo neironu tīklu modelis spēj uzrādīt daudzkārt labākus rezultātus par diferenciālvienādojumos balstīto risinājumu, kļūdas robežās spējot prognozēt nākamās 1.5s, salīdzinot ar 0.075s diferenciālvienādojumos balstītajam risinājumam.

Tomēr mākslīgo neironu tīklu apmācības ilgums ir nesalīdzināmi lielāks par diferenciālvienādojumos balstītā risinājuma aprēķina ilgumu un pieprasa lielus skaitļošanas resursus.

Darba pamattekstā ir 54 lappuses, 82 formulas, 30 attēli, 3 tabulas, 40 izmantotās literatūras avoti.

ABSTRACT

Statistics show that in the last five years the popularity of both deep neural networks altogether, as well as specifically in the field of engineering has risen significantly.

The bachelor's thesis looks at properties of artificial neural networks and practically models a double pendulum system, based on a dataset generated from video recordings. For comparison, a similar model is made by describing the system with differential equations. Actually compared are both models' capabilities of predicting the next 2s of double pendulum motion by using information about the previous second. In addition, both models are compared by their ability to make predictions in specific error margin.

Results show that the best deep artificial neural network model shows much better results than its differential equation based counterpart by being able to predict the next 1.5s of motion in the specified error margin, compared to just 0.075s for the differential equation based model.

But the main drawback of deep artificial neural network models reveals to be that large computational resources are needed for their training.

The main text of the bachelor's thesis consists of 54 pages, 82 equations, 30 images, 3 tables and 40 literature sources.

SATURS

SAĪSINĀJUMI

BN - partijas dimensijas standardizācija (*batch normalization*)
CPU - procesors (*central processing unit*)
FC - lineāras transformācijas slānis (*fully connected layer*)
GAN - ģeneratīvi sacīkstes tīkli (*generative adversarial networks*)
GPU - grafiskais procesors (*graphics processing unit*)
HPC - augstas veiktspējas skaitļošana (*high performance computing*)
LN - slāņa dimensijas standardizācija (*layer normalization*)
LSTM - ilgtermiņa-īstermiņa šūna (*long short-term memory*)
MAE - vidējā absolūtā kļūda (*mean absolute error*)
MLP - daudzslāņu perceptrons (*multi layer perceptron*)
MS - vidējais soļu skaits (*mean step*)
MSE - vidējā absolūtā kļūda (*mean absolute error*)
ODE - parastais diferenciālvienādojums (*ordinary differential equation*)
RNN - rekurentie mākslīgo neironu tīkli (*recurrent neural networks*)

IEVADS

Tipiska pieeja dažādu mehānisku sistēmu aprakstīšanai ir to matemātisko modeļu veidošana - reālais objekts tiek aizstāts ar vienkāršotu matemātisku modeli. Kā alternatīva metode pielietojami datos balstīti modeļi - modelis tiek izveidots apstrādājot eksperimentālus datus no jau eksistējošas sistēmas.

Konkrēta pieeja datu apstrādei ir mākslīgo neironu tīklu pielietošana - pēdējos gados pieejamo skaitļošanas resursu un datu apjoma palielināšanās rezultātā popularitāti guvuši dziļie mākslīgo neironu tīkli. Kā liecina publikāciju datubāzes *Semantic Scholar* dati, publikāciju skaits pēc atslēgvārdiem *deep neural networks* pēdējo 5 gadu laikā pieaudzis no 37 475 publikācijām 2016.gadā līdz 120 714 publikācijām 2021.gadā, kas iezīmē jau pieminēto popularitātes pieaugumu. Līdzīga tendence novērojama arī apskatot tikai inženierzinātņu nozari - tur publikāciju skaits pēdējo 5 gadu laikā pieaudzis no 4052 publikācijām 2016.gadā līdz 8608 publikācijām 2021.gadā.

Daudzi līdzšinējie pētījumi apskata iespējas dziļo neironu tīklu pielietojumiem ar mehāniku saistītu problēmu risināšanai. Kā variants izskatīta diferenciālvienādojumu risināšana - I.Lagaris u.c demonstrējuši veidu, kā ar dziļajiem mākslīgo neironu tīkliem atrisināt parastos un parciāldiferenciālvienādojumus, tādā veidā piedāvājot alternatīvu klasiskajām skaitliskajām metodēm. [1]

Tāpat pētīti pielietojumi, kuros sistēmas tiek modelētas ar laika sēriju datiem paredzētiem dziļajiem neironu tīkliem. J.Klinkahorns u.c. veikuši praktisku salīdzinājumu dažādām mākslīgo neironu tīklu metodēm mehānisku sistēmu aprakstošu parasto diferenciālvienādojumu risināšanai, un tieši laika sērijām paredzētie dziļie neironu tīkli uzrādījuši labākos rezultātus. [**6K2019EvaluatingCM**]. Līdzīgu pētījumu veicis D.Ganons no Indianas Universitātes, pētot dubultā svārsta diferenciālvienādojuma risināšanas iespējas. [2] M.Raissia u.c. piedāvā veidu, kā dziļo neironu tīklu risinājumos iekļaut gan eksperimentālos datus, gan arī zināšanas par sistēmas fizikālajām sakarībām. [3]

Tomēr visiem iepriekš minētajiem pētījumiem ir kopīga īpašība - to pielietotās datu kopas ir ģenerētas, izmantojot tos pašus diferenciālvienādojumus. Pieņemama situācija piemēra demonstrēšanai, bet ne praktiskiem pielietojumiem. Tāpēc informācijas tehnoloģiju uzņēmuma IBM pētnieki A.Assemans u.c. 2019.gada jūlijā publicēja rakstu ”Learning beyond simulated physics” ar kuru kopā tika izdota datu kopa, kas satur datus par dubultā svārsta kinemātiku, ievāktus no reāla svārsta ar video uzņemšanas un apstrādes palīdzību. Šajā pat pētījumā arī apskatīts vienkāršs piemērs, kā prognozēt svārsta kinemātiku ar laika sēriju datiem paredzētām dziļo neironu tīklu metodēm. [4].

Minētās publikācijas iezīmē arī vairākas būtiskas priekšrocības, kuras pielietot praktisku mehānikas problēmu risināšanā. Piem. datu ievākšana bez tieši montētu sensoru palī-

dzības, kas var palīdzēt modelēt ļoti jutīgas sistēmas, kur sensoru pašmasa tās var ietekmēt. Vai arī spēja modelēt nezinot sistēmas parametrus, kas var noderēt diagnostikā. Tālākajā darbā tiks apskatīta jau minētā dubultā svārsta datu kopa un tās modelēšanas iespējas ar dziļo neironu tīklu metodēm, kuru rezultāti tiks salīdzināti ar dubultā svārsta sistēmu aprakstoša diferenciālvienādojuma rezultātiem. Tiks salīdzinātas abu metožu modelēšanas spējas gan īstermiņa, gan ilgtermiņa prognozēm, salīdzināšanas objektivitātei tiks ieviestas speciālas metrikas.

1. DIFERENCIĀLVIENĀDOJUMI

Diferenciālvienādojums ir vienādojums, kas satur kādas funkcijas atvasinājumus, līdz ar to tie ir pielietojami dažādu laikā un telpā mainīgu procesu modelēšanai. Diferenciālvienādojumus, kuri satur viena mainīgā atvasinājumus, sauc par vienkāršiem. Savukārt, ja tie satur vairākargumentu funkciju atvasinājumus, tos sauc par parciālajiem diferenciālvienādojumiem.

Vispārējā gadījumā atrisinot diferenciālvienādojumu tiek iegūta funkciju kopa, kura var aprakstīt jebkuru sākuma stāvokli. Lai iegūtu konkrētu diferenciālvienādojuma atrisinājumu, nepieciešami sākuma nosacījumi - turklāt to skaits sakrīt ar augstākā atvasinājuma pakāpi. Daudzu diferenciālvienādojumu analītiskie atrisinājumi neeksistē vai tie ir ļoti sarežģīti iegūstami, tāpēc praksē diferenciālvienādojumu risināšanai tiek pielietotas skaitliskās metodes, lai atrastu tuvinātus atrisinājumus. [5]

Mehānikā bieži tiek pielietoti kustības diferenciālvienādojumi, kurus iegūst, izmantojot Ņūtona otro likumu. Tie ir pieskaitāmi otrās pakāpes standarta diferenciālvienādojumiem, tāpēc to risināšanai izmantojamās skaitliskās metodes tiks apskatītas tālākajās nodaļās.

1.1. Eilera metode

Pieņem, ka tiek risināts vispārīgs pirmās kārtas parasts diferenciālvienādojums:

$$y' = f(x, y) \tag{1.1}$$

ar sākuma nosacījumu:

$$y(x_0) = y_0 \tag{1.2}$$

Tā atrisinājums tiek meklēts kādā intervālā $[a; b]$, kura sākumpunkts a sakrīt ar x_0. Minētais intervāls tiek sadalīts n vienādās daļās, kur katras daļas garums ir:

$$h = \frac{b - a}{n} \tag{1.3}$$

Eilera metodes pamatā ir funkcijas atvasinājuma ģeometriskā interpretācija - funkcijas grafikam punktā $(x_0, f(x_0))$ vilktās pieskares virziena koeficients sakrīt ar šīs funkcijas atvasinājumu punktā x_0.

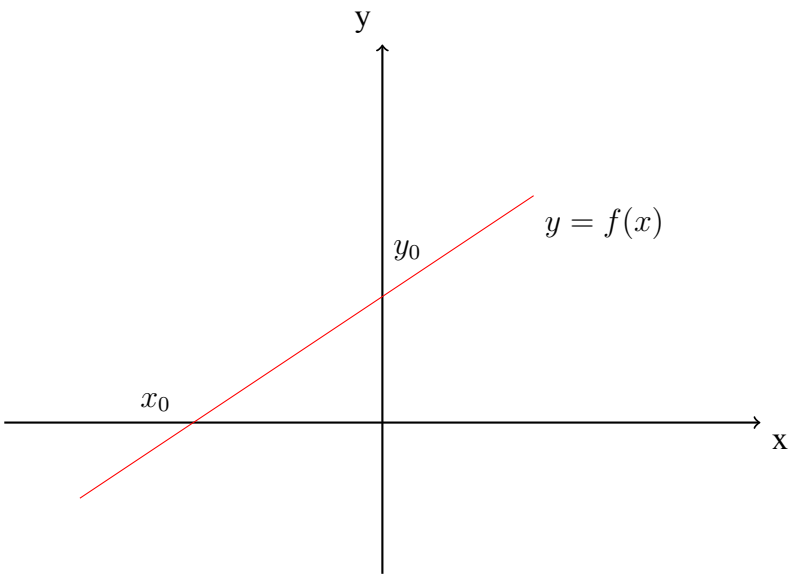
$$k = f'(x_0) \tag{1.4}$$

No analītiskās ģeometrijas zināms, ka taisnes atklātais vienādojums:

$$y = kx + b \tag{1.5}$$

kur: k – virziena koeficients
b – krustpunkts ar y asi

Grafiski raksturīgie punkti attēlojas:



1.1. att. Taisnes raksturīgie punkti

Izmantojot virziena koeficienta definīciju, taisnes atklāto vienādojumu iespējams pārrakstīt:

$$(y_{k+1} - y_k) = k \cdot (x_{k+1} - x_k) \tag{1.6}$$

No vienādojuma izsakot k-to funkcijas vērtību iegūst vispārīgo Eilera metodes vienādojumu:

$$y_{k+1} = y_k + k \cdot (x_{k+1} - x_k) = y_k + f(x_k, y_k) \cdot h \tag{1.7}$$

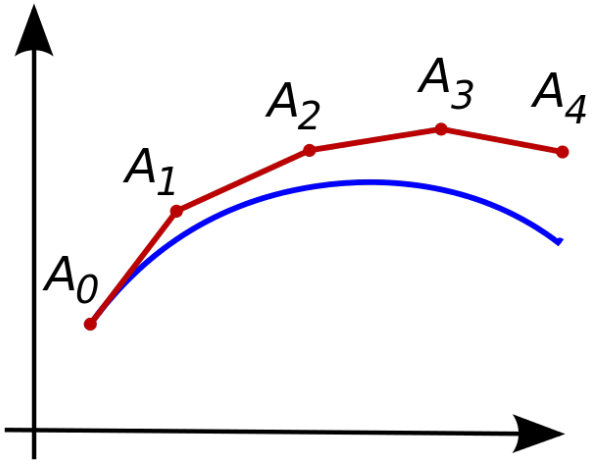
Lai metodi pielietotu praktiski, izmanto skaitliskās diferencēšanas formulu, ar kuru aprēķina funkcijas $f'(x_{k-1},y_{k-1})$ vērtību.

$$y' = \frac{dy}{dt} = \frac{y(t + \Delta t) - y(t)}{\Delta t} = \frac{y(t + h) - y(t)}{h} \tag{1.8}$$

Pielietojot šo vienādojumu sekojošiem režģa punktiem, iegūst Eilera metodes praktiski pielietojamo vienādojumu:

$$y_{k+1} = y_k + f(x_k, y_k) \cdot h = y_k + \frac{y(t + h) - y(t)}{h} \cdot \frac{b - a}{n} \tag{1.9}$$

Kopumā, Eilera metodē rezultātu iegūst, nepārtrauktu funkciju atvasinājumu vērtības aprakstot ar pieskarēm noteiktos, galīgos intervālos. Rezultātā iegūst lauztu līniju



1.2. att. Pieskaru attēlojums funkcijas aproksimēšanā Eilera metodei [6]

Šāda metode ir vienkārša, bet neprecīza, jo izmanto tikai vienu tekošā atvasinājuma vērtību. Tāpēc praksē tiek izmantotas citas metožu grupas, kuras katrā solī koriģē iegūtā atvasinājuma vērtību, rezultējoties augstākā precizitātē. Pie šādām metodēm pieder Runges-Kutta metodes. [5]

1.2. Runges-Kutta metodes

Arī Runges-Kutta metodes darbojas analoģiski Eilera metodes ģeometriskajai interpretācijai.

$$y_{k+1} = y_k + f(x_k, y_k) \cdot h \tag{1.10}$$

Tomēr atšķirībā no Eilera metodes, Runges-Kutta metodes izmanto vairāku virziena koeficientu projekcijas, tādā veidā koriģējot aprēķināmo funkcijas izmaiņas ātrumu. Praksē visbiežāk tiek pielietotas ceturtās kārtas Runges-Kutta metodes, kuras apraksta vienādojumi:

$$y_{k+1} = y_k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \cdot h \tag{1.11}$$

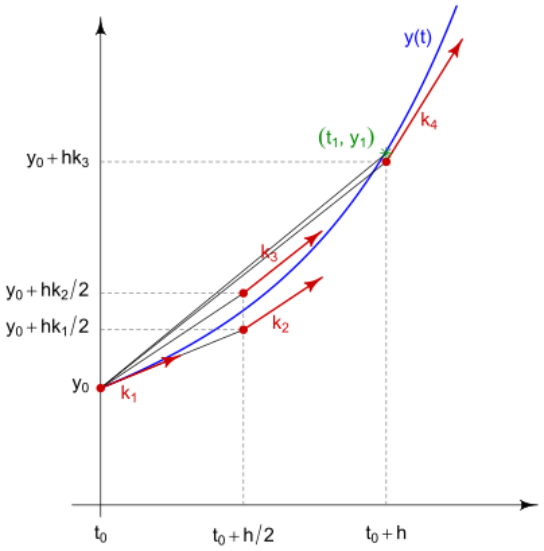
$$k_1 = f(x_k, y_k) \tag{1.12}$$

$$k_2 = f(x_k + \frac{h}{2}, x_n + \frac{h}{2} \cdot k_1) \tag{1.13}$$

$$k_3 = f(x_k + \frac{h}{2}, x_n + \frac{h}{2} \cdot k_2) \tag{1.14}$$

$$k_4 = f(x_k + h, y_k h \cdot k_3) \tag{1.15}$$

Kā novērojams vienādojumos, ceturtās kārtas metode izmanto četras virziena koeficienta projekcijas. [5]



1.3. att. Pieskaru attēlojums funkcijas aproksimēšanā Runges-Kutta metodēm [7]

Lai uzlabotu metožu precizitāti, tiek pielietotas adaptīvās Runges-Kutta metodes, kuras risinājuma gaitā spēj koriģēt algoritma soļa vērtības. Tā vietā lai katrā solī izmantotu noteiktas kārtas Runges-Kutta metodi, tiek izmantotas vairāku kārtu metodes un starp tām veic salīdzinājumu, kā rezultātā tiek iegūta starpība starp prognozētajām vērtībām. Balstoties šajā starpībā tiek pieņemts lēmums, vai nepieciešams koriģēt soļa vērtības:

1. Ja starpība ir lielāka par noteikto pielaides vērtību, tiek samazināts algoritma solis un pārrēķināta starpība, kamēr tā ir zemāka par pielaides vērtību.

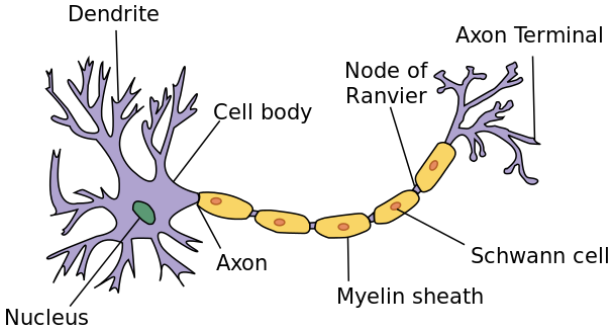
2. Ja starpība ir mazāka par noteikto pielaides vērtību, soli atkarībā no starpības var nemainīt vai palielināt. Palielināšanas gadījumā jārīkojas līdzīgi kā pirmajā gadījumā, veicot iteratīvu starpības salīdzināšanu ar pielaides vērtību

Kā piemēru var minēt vāciešu matemātiķa Ervīna Fēlberga 1969.gadā radīto metodi, kas izmanto ceturtās un piektās pakāpes Runges-Kutta aprēķinu shēmas algoritma soļa adaptīvai koriģēšanai. Mūsdienās to apzīmē kā RKF45 metodi. [8]

2. MĀKSLĪGIE NEIRONU TĪKLI

Mākslīgie neironu tīkli ir no daudziem atsevišķiem elementiem - mākslīgajiem neironiem - sastāvoša skaitļošanas sistēma, kas radīta, iedvesmojoties no cilvēka smadzeņu un nervu sistēmas darbības pamatprincipiem.

Bioloģiskais neirons ir nervu šūna, kas ar elektriskiem signāliem apstrādā un nodod tālāk informāciju. Neirona ķermeni sauc par somu, no tās iziet daudzi izaugumi - dendrīti un aksons. Neironi savā starpā savienoti caur sinapsēm, kuras savieno viena neirona aksonu ar cita neirona aksonu, somu vai dendrītu. Sinapses nodrošina signāla ienākšanu nervu šūnā, bet izejošo signālu aizvada pa aksonu, kurš sazarojas un signālu novada uz daudziem citiem neironiem.



2.1. att. Bioloģiskais neirons [9]

Sinapses savā starpā atšķiras pēc caurlaidības, tāpēc viena stipruma impulsi var dot dažādu kairinājuma pakāpi, nonākot citā neironā. Caur sinapsēm ienākošie signāli dod regulāru kairinājumu, pārsniedzot noteiktu pakāpi arī konkrētais neirons dod izejas signālu uz tālākajiem neironiem, turklāt signāla impulsa stiprums tieši atkarīgs no ieejas signāla impulsa. Tajā pat laikā nepietiekams signāls nerezultējas tālāka impulsa padošanā. [10]

2.1. Neirona matemātiskais modelis

Mākslīgajos neironu tīklos tiek izmantoti neironu matemātiskie modeļi, kuri atdarina atsevišķas bioloģisko neironu iezīmes. Turpmākajā tekstā neirons tiks izmantots tieši šajā nozīmē. Neirons ir vienkāršs skaitļošanas elements, kuram ir daudzas ieejas un viena izeja un kurš elektrisko signālu vietā saņem skalāras vērtības. Tipiski izmantotais neirons sastāv no:

1. Svariem(*weights*) un nobīdes vērtībām(*bias*)
2. Summēšanas funkcijas
3. Aktivizācijas funkcijas
4. Izejas funkcijas

Neironam darbojoties uz tā ieejām tiek padotas vērtības, kuras summēšanas funkcija, izmantojot neirona svarus, pārveido vienā vērtībā. Visbiežāk izmantotā summēšanas funkcija parādīta formulā (2.1), kurā katrs individuālais signāls tiek reizināts ar attiecīgo svaru un tiek pieskaitīta nobīdes vērtība (*bias*).

$$\sum_{i=1}^{k} w_i x_i + \theta \tag{2.1}$$

Tālāk šī summas vērtība tiek padota uz aktivizācijas un izejas funkcijām.

$$\mathcal{F}(\sum_{i=1}^{k} w_i x_i + \theta) \tag{2.2}$$

Visbeidzot izejas funkcija no aktivizācijas funkcijas dotās vērtības aprēķina neirona izejas vērtību. [10]

2.2. Aktivācijas funkcijas

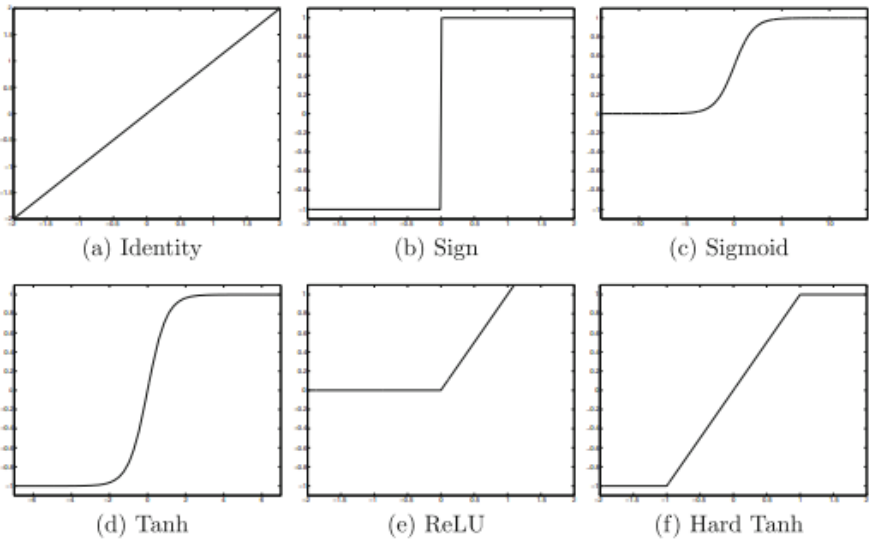
Aktivācijas funkcijas loma ir apstrādāt summēšanas funkcijas izejas vērtības. Tās var būt lineāras vai nelineāras, taču tieši nelinearitātes īpašība ir svarīga vairāku kārtu (*multi-layer*) neironu tīklos. Klasiski izmantotās aktivācijas funkcijas ietver identitātes funkciju, sigmoīdu, hiperbolisko tangensu.

$$\mathcal{F}(x) = x \tag{2.3}$$

$$\mathcal{F}(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

$$\mathcal{F}(x) = tanh(x) \tag{2.5}$$

Pēdējos gados lielāku uzmanību izpelnījušās ReLU jeb max(0,x) un tā variācijas zemāka aprēķina kompleksuma dēļ. Arī praktiski pētījumi liecina par to efektivitāti. [11] [12]

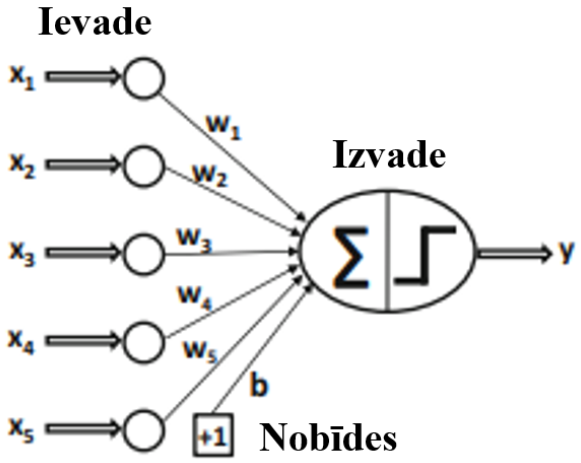


2.2. att. Aktivācijas funkcijas [12]

Attēls 2.2. ilustrē minēto aktivācijas funkciju izejas vērtības atkarībā no ieejas vērtībām.

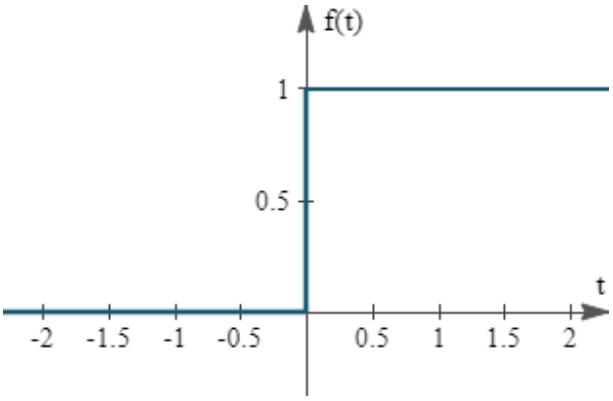
2.3. Perceptrons

Kā viens no pirmajiem mākslīgo neironu tīklu modeļiem tika radīts perceptrons, kuru 1958.gadā izveidoja Masačūsetsas Tehnoloģiju institūta pētnieks F.Rozenblats.



2.3. att. Perceptrons [12]

Perceptrona modelis tika izstrādāts bināras klasifikācijas veikšanai un tas kā aktivācijas funkciju izmantoja Hevisaida funkciju.



$$H(x) = \begin{cases} 0, \text{ ja } x \leq 0 \\ 1, \text{ ja } x > 0 \end{cases} \tag{2.6}$$

2.4. att. Hevisaida funkcija

Apkopojot iepriekš apskatīto vispārīgo mākslīgā neirona modeli, algebriski perceptronu izsaka:

$$izeja = \begin{cases} 0 & \text{ja } \sum_{i=1}^{k}(w_i x_i) + \theta \leq 0 \\ 1, & \text{ja } \sum_{i=1}^{k}(w_i x_i) + \theta > 0 \end{cases} \tag{2.7}$$

Tomēr izmantojot dažādas aktivācijas funkcijas mākslīgo neironu tīklu modeļi var veikt gan klasifikācijas, gan regresijas uzdevumus. [13]

2.4. Mākslīgais neironu tīkls

Vispārīgā gadījumā par mākslīgo neironu tīklu sauc skaitļošanas sistēmu, kura sastāv no liela daudzuma jau iepriekš apskatīto skaitļošanas elementu - mākslīgo neironu. Liels neironu skaits ļauj izveidot dažādus to izkārtojumus, kurus raksturo neironu tīkla arhitektūra. Tāpat neironu tīklus atsevišķi pēc slāņu savstarpējās sadarbības iedala:

- Vienvirziena tīklos (*Feedforward networks*)
- Rekurentos tīklos (*Recurrent/Feedback Networks*)

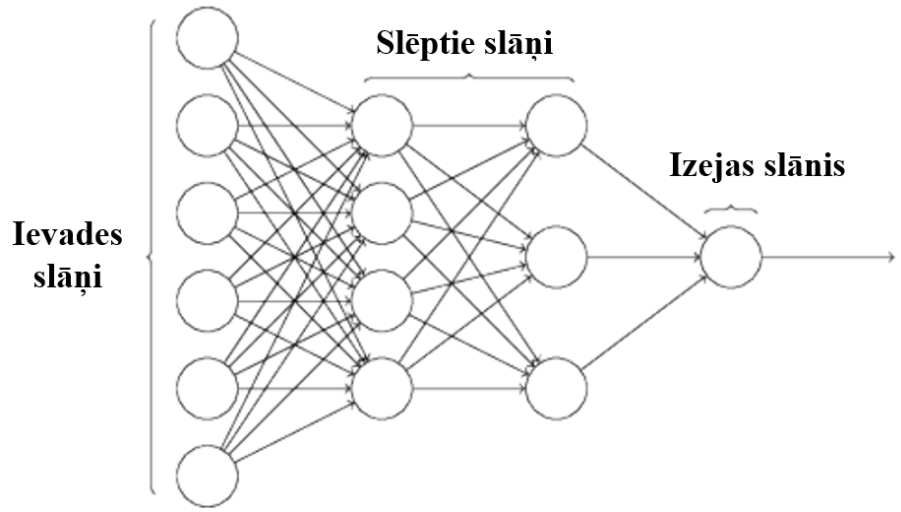
Galvenā atšķirība starp tiem ir savienojumos starp neironiem dažādos slāņos - ja vienvirziena tīklos neirona izeja savienojas tikai ar citiem neironiem vienā virzienā (pa slāņiem uz priekšu), tad rekurentajos tīklos pieļaujamas arī saites uz neironiem otrā virzienā vai pašam neironam uz sevi. [10]

2.5. Daudzslāņu perceptrons

Arī savienojot daudzus iepriekš apskatītos perceptronus var iegūt viena slāņa mākslīgo neironu tīklu, tomēr tas praktiski netiek pielietots, jo nespēj risināt nelineārus uzdevumus. Šo problēmu var atrisināt ar vairāku slāņu neironu tīkliem. Tipiski pielietots modelis ir daudzslāņu perceptrons MLP (*multy layer perceptron*), uz kura bāzes tiks balstīts tālākais apraksts. Jāņem vērā, ka apzīmējums MLP nenozīmē, ka tas obligāti sastāv no perceptrona modeļa mākslīgajiem neironiem, bet ar to parasti tiek apzīmēts jebkurš daudzslāņu vienvirziena neironu tīkls. Daudzslāņu perceptrons tipiski sastāv no sekojošiem slāņiem:

- Ieejas slāņa, kurš saņem ieejas vērtības.
- Izejas slāņa, kurš dod neirona tīkla galējās izejas vērtības.
- Vismaz viena slēptā slāņa (*hidden layer*), kuru neironu ievades un izvades ir tieši savienotas ar neironiem citos slāņos.

Katrs atsevišķais mākslīgais neirons izmanto nelineāru aktivizācijas funkciju, variācijas iespējamas tikai pēdējā slānī atkarībā no veicamā uzdevuma specifikas.



2.5. att. Neironu tīkls ar diviem slēptajiem slāņiem [13]

Tas tiek darbināts sekojoši pa slāņiem - katra neirona izeja ir savienota ar nākamā slāņa katra neirona ieeju(ja vien tas nav pēdējais slānis, kura izeja arī ir galējā izejas vērtība).[10]

2.6. Mākslīgo neironu tīklu apmācība

Vispārīgā veidā mākslīgo neironu tīklu mērķis ir aproksimēt kādu funkciju f. Lai to paveiktu, jāveic tīkla apmācība. Kā tika noskaidrots apakšnodaļā 2.1, neironiem ir parametri

- svari un nobīdes vērtības. Tieši šo vērtību pielāgošana saprotama kā mākslīgā neironu tīkla apmācība. Lai to praktiski īstenotu vajadzīgi algoritmi, kuri:

- Ļauj iteratīvi koriģēt tīkla parametrus.
- Nosaka koriģēšanas apmēru un virzienu.

Apmācības procesa gaitā nepieciešama kļūdas funkcija(*loss function*). Kļūdas funkcijas atšķiras atkarībā no uzdevuma tipa, taču to kopīgais mērķis ir salīdzināt tīkla izejas vērtības ar aproksimējamās funkcijas vertībām. Piem. regresijas uzdevumos bieži tiek izmantota MSE(*Mean-square error*) kļūdas funkcija.

$$\mathcal{L} = \frac{1}{n}\sum(y(x) - a)^2 \tag{2.8}$$

kur: $y(x)$ – funkcijas vērtība
a – tīkla izejas vērtība
n – ieejas vērtību skaits

Rezultātā var secināt, ka mākslīgo neironu tīklu apmācībā būtībā ir kļūdas funkcijas minimizēšana. Konkrēti daudzslāņu perceptronu modeļos parametru koriģēšanai tiek izmantots atpakaļ-izplatīšanās algoritms (*Backpropagation*) un kalnākāpēja(*Gradient Descent*) algoritms. [13]

2.7. Kalnākāpēja algoritms

Pieņem, ka tiek minimizēta kāda vairākargumentu funkcija.

$$f(x_1, x_2, ..., x_n) \tag{2.9}$$

Funkcijas pieagums kādā punktā vektora x virzienā nosakāms kā:

$$\Delta f \approx \frac{\partial f}{\partial x_1}\Delta x_1 + ... + \frac{\partial f}{\partial x_n}\Delta x_n \tag{2.10}$$

Kā zināms, funkcijas gradients kādā punktā nosaka virzienu, kurā funkcija aug visātrāk.

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ ... \\ \frac{\partial f}{\partial x_n} \end{bmatrix}) \tag{2.11}$$

Tātad funkcijas pieaugumu iespējams pārrakstīt:

$$\Delta f \approx \nabla f \cdot \Delta x \tag{2.12}$$

Tā kā mērķis ir funkciju minimizēt, jāizvēlas virziens tieši pretējs gradientam. Tad izmaiņas parametros var izteikt kā:

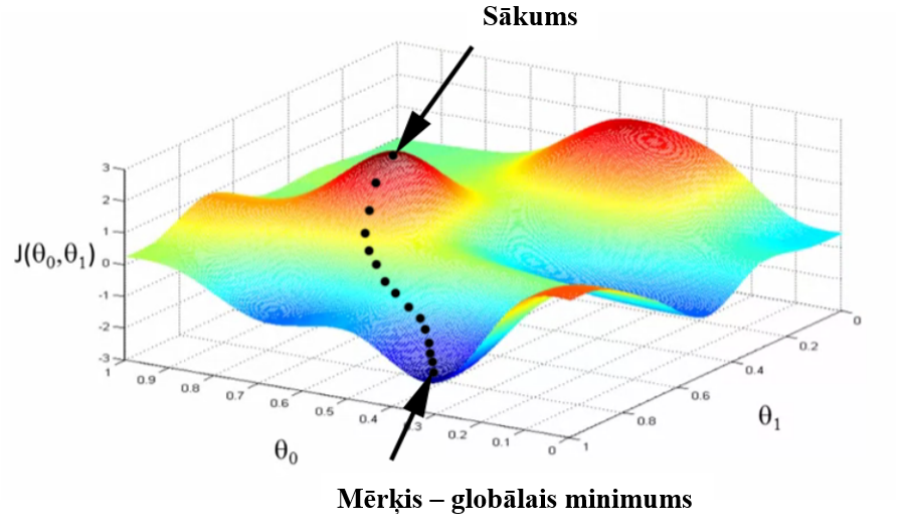
$$\Delta x = -\eta \cdot \nabla f \tag{2.13}$$

kur: $\eta -$ apmācības ātrums

Praktiski tiek panākts, ka parametru izmaiņa tiks veikta tā, lai samazinātu kopējo funkcijas vērtību. Apmācības ātrums formulā ir neliela vērtība, un tā vajadzīga, lai samazinātu kopējo parametru izmaiņas soļa lielumu. Pretējā gadījumā parametri var tikt izmainīti par daudz un funkcijas vērtība nevis tuvosies, bet gan tiks nobīdīta tālāk no globālā minimuma. Ievietojot iegūto parametru izmaiņas vērtību sākotnējā vienādojumā, iegūst:

$$\Delta f \approx -\eta \nabla f \cdot \nabla f = -\eta (\nabla f)^2 \tag{2.14}$$

Tā kā $(\nabla f)^2 \geq 0$, tad $\Delta f \leq 0$, kas nozīmē, ka funkcijas vērtība samazināsies katrā solī. Praktiski algoritma darbības vizualizāciju konkrētai funkcijai var novērot attēlā:



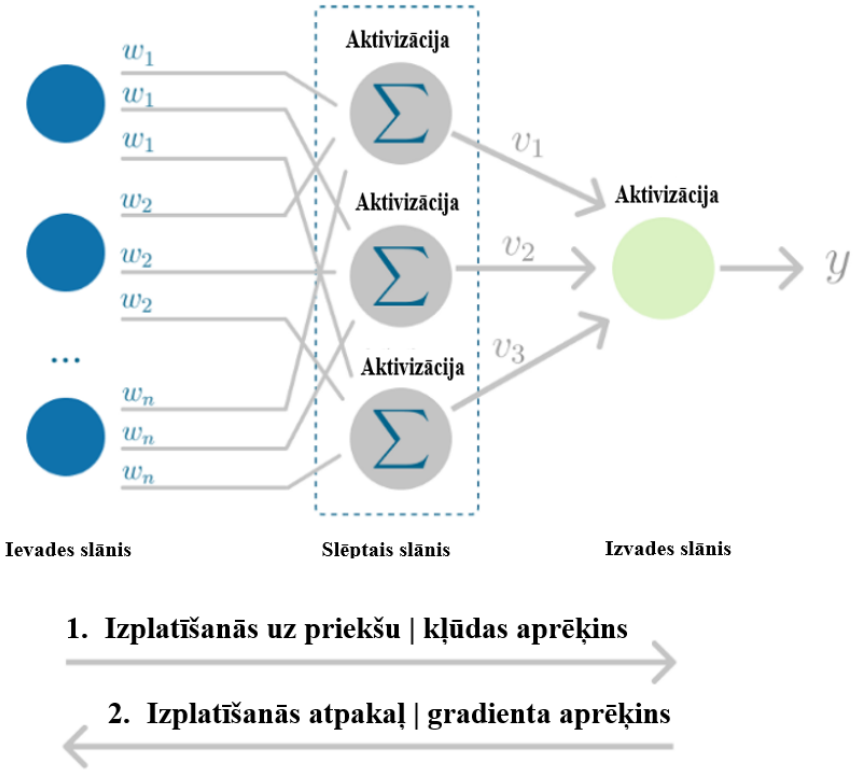
2.6. att. Divargumentu funkcijas grafiks telpā un algoritma ceļš [14]

Kā redzams attēlā, konkrētā funkcijas vērtība samazinās ar katru iterāciju. Tātad šo algoritmu iespējams izmantot arī mākslīgo neironu tīkla kļūdas funkcijas minimizēšanai. Bet tādā gadījumā vajadzīgs arī algoritms, kurš spēj noteikt gradientu. [13]

2.8. Atpakaļ izplatīšanās algoritms

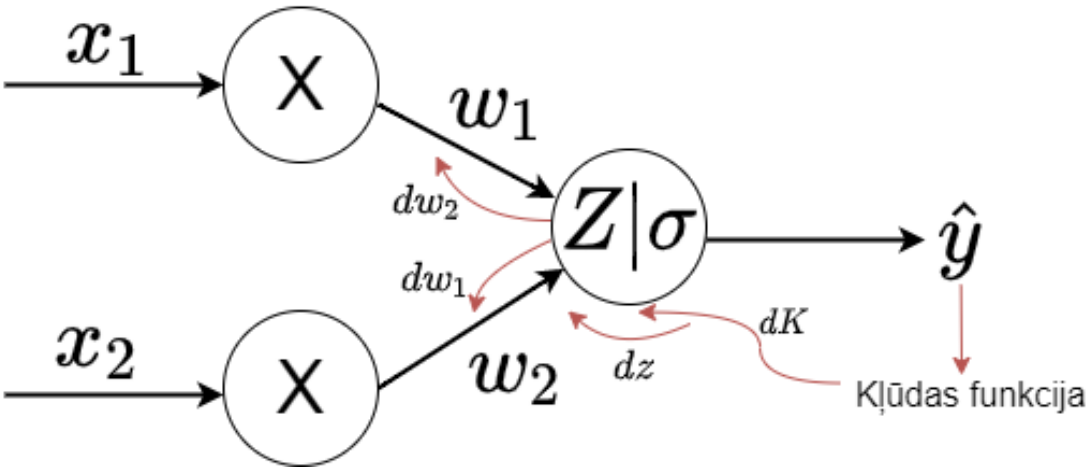
Atpakaļ izplatīšanās algoritms spēj aprēķināt parciālos atvasinājumus starp visiem ievades un izvades vērtību pāriem, iegūstot gradientu. Tad izmantojot kalnākāpēja algoritmu

tīkla parametri tiek koriģēti. Ilustratīvs piemērs parādīts 2.7.att.:



2.7. att. Mākslīgā neironu tīkla apmācības cikls [15]

Praktiskam apmācības piemēram tiks izmantots 2.8.att. ilustrētais vienkāršais modelis ar divām ieejas vērtībām un vienu izejas vērtību.



2.8. att. Piemērā izmantotais mākslīgo neironu tīkls

Tīklā izmantotie apzīmējumi:

- x_n - n-tā ieejas vērtība
- w_n - n-tais svars
- z - summēšanas funkcija
- σ - aktivizācijas funkcija
- $\bar{y}$ - tīkla izejas vērtība

Darbības sākumā svari ir jāinicializē, parasti izmanto nejaušas vērtības. Kā iepriekš tika noskaidrots, izejas vērtības iegūst, darbinot tīklu sekojoši pa slāņiem uz priekšu. Tātad šajā gadījumā izejas vērtības pakāpeniski iegūst algebriskā formā:

$$z = \sum_{i=1}^{n} w_n x_n + b = w_1 x_1 + w_2 x_2 + b \tag{2.15}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.16}$$

$$\bar{y} = \sigma(\sum_{i=1}^{n} w_n x_n) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + b)}} \tag{2.17}$$

Kad iegūta izejas vērtība, tiek aprēķināta kļūdas funkcijas vērtība. Šajā piemērā kā kļūdas funkcija tiks izmantota vidējā kvadrātiskā kļūda.

$$\mathcal{L} = MSE = \frac{1}{n} \sum_{i=1}^{n} (\bar{y} - y)^2 = \frac{1}{2} (\bar{y} - y)^2 \tag{2.18}$$

kur: $\bar{y}$ – tīkla aprēķinātās izejas vērtības
y – reālās datu vērtības

Pēc kļūdas iegūšanas jāveic svaru koriģēšana. Šeit tiek izmantots atpakaļ-izplatīšanās algoritms - tiek iets pa slāņiem uz atpakaļu, pakāpeniski iegūstot kļūdas funkcijas parciālos atvasinājumus pret katru no svariem. Praktiski šī operācija ir ekvivalenta saliktas funkcijas atvasināšanai.

$$\frac{\partial \mathcal{L}(\bar{y},y)}{\partial w_1} = \frac{\partial \mathcal{L}(\bar{y},y)}{\partial (\bar{y} - y)} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{w_1} \tag{2.19}$$

Kļūdas funkcijas atvasinājums:

$$\frac{\partial \mathcal{L}(\bar{y},y)}{\partial (\bar{y} - y)} = 2 \cdot (\bar{y} - y) \cdot \frac{1}{2} = (\bar{y} - y) \tag{2.20}$$

Sigmoīda atvasinājums:

$$\frac{d\sigma}{dz} = \sigma(z) \cdot (1 - \sigma(z)) \tag{2.21}$$

Summēšanas funkcijas atvasinājums:

$$\frac{d(w_1 x_1 + w_2 x_2 + b)}{dw_1} = x_1 \tag{2.22}$$

Rezultātā iegūst:

$$\frac{\partial \mathcal{L}(\bar{y},y)}{\partial w_1} = (\bar{y} - y) \cdot \sigma(z) \cdot (1 - \sigma(z)) \cdot x_1 \tag{2.23}$$

Pēdējā izteiksme arī izsaka svara w_1 kļūdas funkcijas parciālo atvasinājumu pret attiecīgo svaru. Līdzīgā veidā iegūst arī parciālos atvasinājumus pēc otra svara un nobīdes vērtības

$$\frac{\partial \mathcal{L}(\bar{y},y)}{\partial w_1} = 2(\bar{y} - y) \cdot \sigma(z) \cdot (1 - \sigma(z)) \cdot x_1 \tag{2.24}$$

$$\frac{\partial \mathcal{L}(\bar{y},y)}{\partial w_2} = 2(\bar{y} - y) \cdot \sigma(z) \cdot (1 - \sigma(z)) \cdot x_2 \tag{2.25}$$

$$\frac{\partial \mathcal{L}(\bar{y},y)}{\partial b} = 2(\bar{y} - y) \cdot \sigma(z) \cdot (1 - \sigma(z)) \tag{2.26}$$

Arī sarežģītākiem modeļiem ar daudziem slēptajiem slāņiem aprēķina princips nemainās. Kad visi parciālie atvasinājumi atrasti, ar kalnākāpēja algoritmu jāatrod jaunās svaru un nobīdes vērtības, kas rezultātā minimizēs kļūdas funkciju.

$$\bar{w_1} = w_1 - \eta \cdot \frac{\partial \mathcal{L}(\bar{y},y)}{\partial w_1} \tag{2.27}$$

$$\bar{w_2} = w_2 - \eta \cdot \frac{\partial \mathcal{L}(\bar{y},y)}{\partial w_2} \tag{2.28}$$

$$\bar{b} = b - \eta \cdot \frac{\partial \mathcal{L}(\bar{y},y)}{\partial b} \tag{2.29}$$

Praksē pārsvarā tiek izmantots stohastiskais kalnākāpēja algoritms - svaru koriģēšanai tiek izmantota nevis visa datu kopa uzreiz, bet gan nejauša apakškopa, kas satur tikai daļu no kopējo datu apjoma. Šim nolūkam datu kopu sadala partijās (*batches*) ar konkrētu izmēru un apmācības laikā pakāpeniski laiž cauri modelim.

Tāpat svarīgs jēdziens mākslīgo neironu tīklu apmācības procesā ir epohas - apmācības cikli, kuru laikā cauri mākslīgo neironu tīklam izlaisti visi pieejamie datu punkti un veikta attiecīgā svaru pielāgošana. Parasti mākslīgo neironu tīkli tiek apmācīti pietiekamu epohu skaitu, kamēr kļūdas funkcija ir konverģējusi - pēdējā iterācijā tā ir mainījusies par vērtību, kas ir zemāka par kādu noteiktu robežvērtību vai pat palielinājusies. [12]

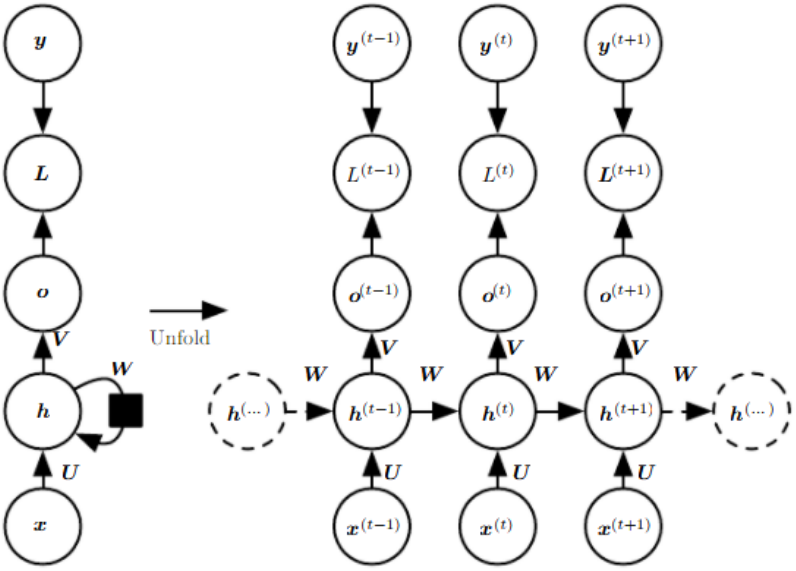
Apmācības procesā svarīgi ņemt vērā arī skaitļošanas īpašības. Datori informācijas

apstrādi veic centrālajā procesorā CPU, taču tas vienlaicīgi spēj veikt tikai vienu darbību un neironu tīklu apmācība tādā veidā ir neefektīva. Lai uzlabotu mākslīgo neironu tīklu veiktspēju, tiek izmantota paralēlapstrāde - datoriekārtas ar spēju paralēli apstrādāt vairāk kā vienu informācijas vienību. Šādu iespēju piedāvā grafiskie procesori GPU (*graphics processing units*) un tieši šī iemesla dēļ tie kļuvuši par piemērotu risinājumu lielu dziļo mākslīgo neironu tīklu apmācības procesā. [16]

2.9. Rekurentie neironu tīkli

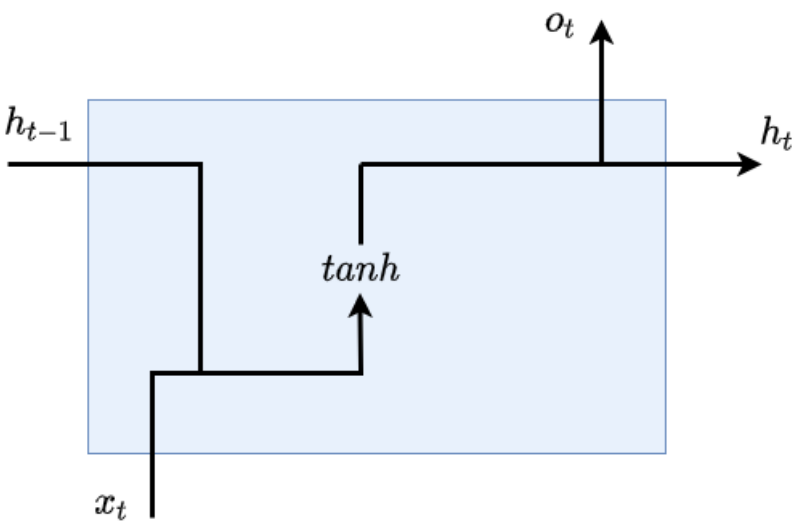
Atšķirībā no iepriekšējā nodaļā apskatītajiem vienvirziena neironu tīkliem, rekurentajos neironu tīklos signāls neironos tiek virzīts ne tikai pa slāņiem uz priekšu, bet arī atpakaļ neironiem pašiem uz sevi.

Rezultātā iegūtais neironu tīkls gūst iespēju apstrādāt laika sēriju datus - ja vienvirziena tīklā ieejas vērtības bija tikai datu punktu vērtības, tad rekurentajos neironu tīklos ieejas vērtības ir gan datu punktu vērtības, gan to secība. Tātad katra neirona ieejas signāls ir atkarīgs ne tikai no tiešā ieejošā signāla, bet arī no visiem iepriekšējiem ieejošajiem signāliem.[17]



2.9. att. Rekurentā mākslīgo neironu tīkla shēma [17]

Šāda neironu tīkla arhitektūra ir noderīga modelējot datus, kuru īpašības nosaka arī to secība, kā valodas modelēšana, runas atpazīšana, tulkošana u.c. Tālāk apskatīts rekurento mākslīgo neironu tīklu pamatelements.



2.10. att. Rekurentā mākslīgo neironu tīkla atsevišķa elementa shēma

Kā redzams 2.10.att. atsevišķā elementa shēmā ir divas ieejas un divas izejas vērtības:

- h_{t-1} - slēptais kanāls, kurš satur informāciju par iepriekšējām ievades vērtībām.
- x_t - kārtējā ievades vērtība.
- h_t - atjaunotā slēptā kanāla vērtība.
- o_t - elementa izejas vērtība.

Tieši slēptais kanāls nodrošina spēju 'atcerēties' iepriekšējos ievades datus, kas ļauj modelēt secīgus notikumus. Šis slēptais kanāls katrā elementā tiek atjaunots ar jaunākajām ievades vērtībām, algebriski tā ir funkcija no iepriekšējā signāla un ienākošā signāla.

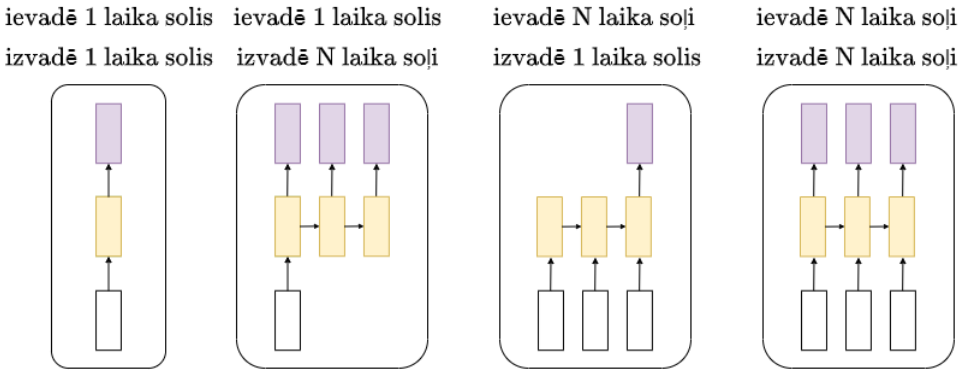
$$h_t = f(h_{t-1}, x_t) \tag{2.30}$$

Balstoties uz 2.10.att. algebriski šo arhitektūru var aprakstīt:

$$h_0 = 0 \tag{2.31}$$
$$x_t = W h_{t-1} + U x_t + b \tag{2.32}$$
$$h_t = \sigma(x_t) \tag{2.33}$$
$$o_t = c + V h_t \tag{2.34}$$
$$\bar{y_t} = \sigma(o_t) \tag{2.35}$$

Kur kā parametri b un c pieņemti attiecīgo slāņu nobīdes un U, V, W kā svari. [17]

Kā redzams, slēptais kanāls h_t katrā iterācijā tiek koriģēts ar ienākošā signāla vērtībām, pēc tam tāda pati operācija tiek atkārtota nākamajā slānī u.t.t. Tāpat arī iespējamas dažādas variācijas signāla izplatīšanā, jo var mainīt gan ieejas, gan izejas vērtību sēriju garumus un izejas vērtības aprēķina biežumu.



2.11. att. Iespējamie rekurento mākslīgo neironu tīklu darbības veidi

Piem. modelējot valodu 2.11.att. ilustrētajam gadījumam ar N laika soļiem ievadē un izvadē atbilst situācija tulkošanā, kur no ieejas teikuma tiek ģenerēts tulkotais teikums. Turpretim situācija ar N ievades laika soļiem un 1 izvades laika soli izpildītos, kad ieejas vērtības būtu daudzi vārdi, bet gala izejas vertība būtu tikai viena - piem. teikums tiek klasificēts pēc stila, tad ieejas vērtības ir visi teikuma vārdi, bet izejas vērtība ir viens noteiktais stils.

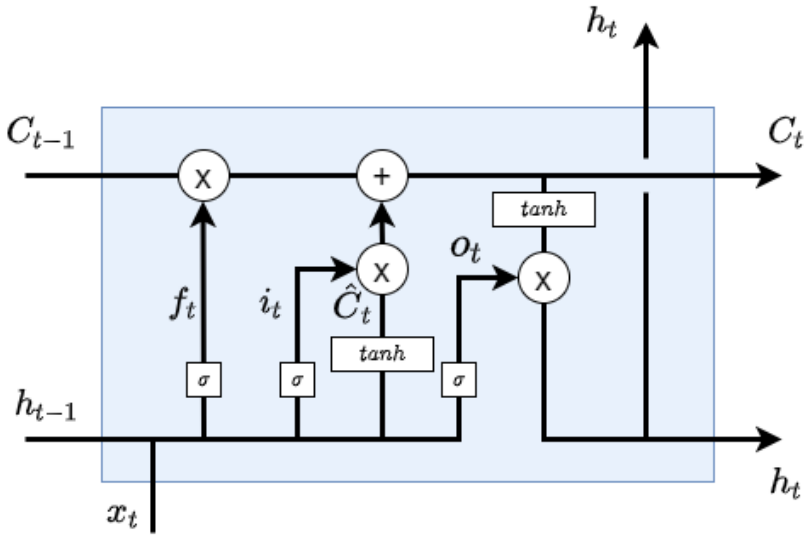
Tomēr šādu modeļu pielietošanai ir problēmas - rekurentajos neironu tīklos atpakaļizplatīšanās algoritmam jādarbojas pāri ne tikai ienākošajam signālam, bet arī visiem iepriekšējiem, jo tie kā slēptā kanāla vērtības piedalās katra soļa aprēķinā. Taču šāda pieeja ir neefektīva, jo ar katru ienākošo datu soli palielinās aprēķina laiks, kā arī gradienta vērtības ņemot vērā visus iepriekšējos soļus var kļūt izzūdošas (*vanishing gradients*) vai tiekties uz bezgalību (*exploding gradients*). [17]

Praktiski tas nozīmē, ka var rasties problēmas ar savstarpēji saistītu datu noteikšanu, ja tie atrodas lielos intervālos viens no otra. Viens no praksē lietotajiem modeļiem, kas šīs problēmas cenšas risināt, ir LSTM.

2.10. LSTM

LSTM (*Long short-term memory*) jeb ilgtermiņa-īstermiņa modelis ir Zepa Hohreitera un Jirgena Šmīdhūbera publikācijā ierosināts modelis, kurš spētu nevis atcerēties visus iepriekšējos soļus, bet iegaumēt tikai svarīgākos, tādā veidā novēršot izzūdošo gradientu problēmu. Tas tiek panākts ar speciālu iekšējā stāvokļa vektoru c_t , kura stāvoklis tiek regulēts ar trim savienojumiem - ieejas, izejas un aizmiršanas vārtiem. [18]

Tipiska LSTM šūna redzama 2.12.att.



2.12. att. LSTM šūnas shēma

Šūnā ieiet ieejas vektors x_t un no iepriekšējiem slāņiem iegūtais stāvokļa vektors c_{t-1} un slēptā stāvokļa vektors h_{t-1}, izejas vektori ir atjaunotie stāvokļa un slēptā stāvokļa vektori c_{t-1} un h_{t-1}. Kā redzams shēmas augšpusē, stāvokļa vektors tiek modificēts divas reizes:

1. Aizmiršanas vārti f_t nosaka, kādas vērtības jāaizmirst. Tie ņem vērā ieejas vērtības x_t un slēptā slāņa h_t vērtības un ar svariem, nobīdes vērtībām un sigmoīda palīdzību iegūst vērtības intervālā [0,1], kuras piereizinot stāvokļa vektoram samazina attiecīgās vērtības, praktiski liekot aizmirst.

$$f_t = \sigma(W_f \cdot [h_{t-1,x_t}] + b_f) \tag{2.36}$$

2. Otra modifikācija nosaka, kāda informācija stāvokļa vektoram jāpievieno. Ievades vārti i_t nosaka, kuras vērtības tiks mainītas, bet atsevišķa hiperboliskā tangensa funkcija *tanh* nosaka konkrētas pievienojamās vērtības. Abu vērtību reizinājums tiek pieskaitīts stāvokļa vektoram c_t, praktiski liekot iegaumēt jaunu informāciju.

$$i_t = \sigma(W_i \cdot [h_{t-1},x_t] + b_i) \tag{2.37}$$
$$\hat{C}_t = tanh(W_c \cdot [h_{t-1},x_t] + b_C) \tag{2.38}$$

Rezultātā stāvokļa vektors tiek atjaunots.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t \tag{2.39}$$

Kā pēdējā operācija tiek aprēķināta šūnas izejas vērtība. Datu izejas vārti o_t, līdzīgi kā ievades un aizmiršanas vārti, iegūst vērtības, kuras tiek sareizinātas ar stāvokļa vektora c_t izeju un rezultāts ir jaunā šūnas izejas vērtība.

$$o_t = \sigma(W_o \cdot [h_{t-1},x_t] + b_o) \tag{2.40}$$

$$h_t = o_t * tanh(C_t) \tag{2.41}$$

Tipiski LSTM šūnas izmanto vairākos slāņos, līdz ar to tālākās šūnas kā ievades vērtības saņems arī no iepriekšējām šūnām nākošos stāvokļa un slēptā stāvokļa vektorus. [19]

2.11. Normalizācija un standardizācija

Apmācot mākslīgo neironu tīklus ir svarīgi, lai ievades datu vērtības būtu savā starpā vienādos mērogos un līdzīgi izkliedētas. Lai datiem piešķirtu vienādu mērogu pielieto normalizāciju.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.42}$$

kur: x' – normalizētā vērtība
x_{min} – datu kopas minimālā vērtība
x_{max} – datu kopas maksimālā vērtība

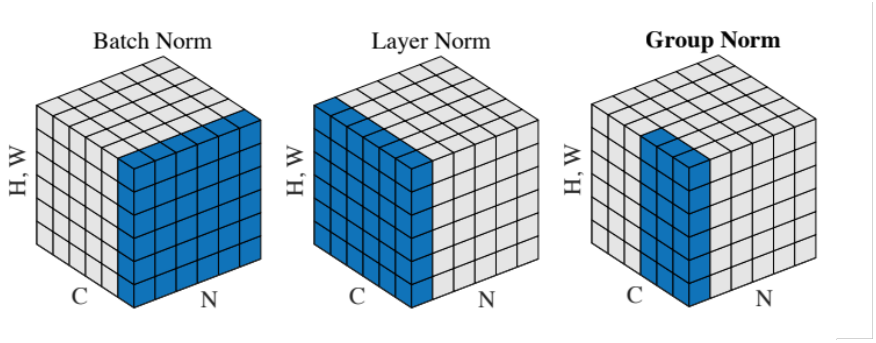
Formula (2.42) datus pārvērš mērogā [0,1]. Līdzīgi datiem var veikt standardizāciju, kuras rezultātā datu kopas vidējā vērtība kļūst par 0 un standartnovirze ir 1.

$$x_S = \frac{x - \mu}{\sigma} \tag{2.43}$$

kur: x_S – standardizētā vērtība
μ – datu kopas vidējā vērtība
σ – datu kopas standartnovirze

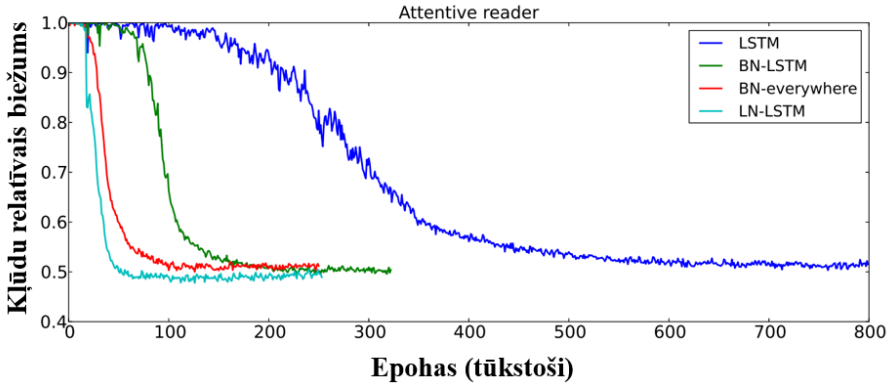
Tomēr pētījumi parādījuši, ka standardizācija arī neironu tīkla iekšienē var dod uzlabojumus. Apmācot ļoti dziļus mākslīgo neironu tīklus rodas problēma - atpakaļizplatīšanās algoritmam nosakot gradientu specifiskā slānī netiek ņemtas vērā tālāku slāņu svaru izmaiņas, bet patiesībā tiek mainīti svari visos slāņos. Rezultātā apmācība ir lēna, neefektīva un grūti sasniegt konverģenci.[17]

Problēmas risināšanai radīti vairāki standardizācijas algoritmi, kurus pielietot neironu tīkla iekšienē, lai novērstu minētās problēmas ietekmi.



2.13. att. Neironu tīklu iekšējo standardizāciju paveidi atkārībā no darbības dimensijas [20]

Kā redzams 2.13.att., minētās standardizācijas metodes darbojas pāri partijas (*batch*), slāņa (*layer*) vai jauktām (*group*) dimensijām. Visas trīs metodes atkarībā no uzdevuma tipa spējušas dot krasu uzlabojumu dziļo neironu tīklu apmācības ātrumā. [21] [22] [23] Rekurento neironu tīklu uzdevumiem tipiski pielietota slāņa dimensijas standardizācija *Layer Norm*, kas arī uzrādījusi labus rezultātus testos no oriģinālās publikācijas.



2.14. att. Slāņa un partijas standardizācijas pielietojuma rezultāti RNN tipa uzdevumā - relatīvais kļūdas biežums atkarībā no epohas [22]

2.14.att. ilustrētajā eksperimentā slāņa standardizācija ļāva par 60% samazināt konverģences sasniegšanai nepieciešamo laiku valodas modelēšanas uzdevumā, turklāt arī uzrādot labāku rezultātu par partijas standardizācijas pieeju. [22]

3. SISTEMĀTISKĀ LITERATŪRAS ANALĪZE

Šajā nodaļā apskatītas citu autoru radītas pieejas un metodes līdzīgu uzdevumu risināšanā. Galvenā uzmanība vērsta dziļo neironu tīklos balstītām metodēm, pētot gan konkrētus risinājumus dubultā svārsta modelēšanai, gan apskatot vispārējas metodes dinamisku sistēmu, kā arī jebkādu citu datu sēriju modelēšanai citās nozarēs. Pētāmie jautājumi:

- Kādas dziļajos neironu tīklos balstītas metodes pielietotas dubultā svārsta sistēmas modelēšanā? Vai veikts salīdzinājums starp cita tipa metodēm?
- Kādas dziļajos neironu tīklos balstītas metodes pielietotas citu dinamisku sistēmu analizēšanai?
- Kādas pieejas izmantotas ilgtermiņa datu sēriju prognožu uzlabošanai?

Publikāciju meklēšanai kā atslēgvārdi izmantotas frāzes: *data-driven dynamical systems, double pendulum deep learning, time-series modelling, differential equation solving neural networks*. Atrastās publikācijas tiek analizētas un kādam no pētāmajiem jautājumiem atbilstošās apkopotas tabulās, rindotas pēc to publikācijas datumiem augošā secībā, kā arī to numuri saglabāti visās tabulās. Pirmajā tabulā apkopota vispārēja informācija par publikāciju autoriem, to pārstāvētajām iestādēm, valsti un publikācijas laiku.

3.1. tabula.

Autori un publikācijas saistībā ar dziļo neironu tīklu metodēm.

Nr.	Nosaukums	Autori	Iestāde	Valsts	Gads
1	Artificial Neural Networks for Solving Ordinary and Partial Differential Equations [1]	I.Lagaris, A.Likas, D.Fotiadis	Jonīnas universitāte	Grieķija	1998
2	Professor Forcing: A New Algorithm for Training Recurrent Networks [24]	A.Goyal, A.Lamb, Y.Zhang, S.Zhang, A.C.Courville, Y.Bengio	Montreālas Universitāte	Kanāda	2016
3	Automated Curriculum Learning for Neural Networks [25]	A.Graves, M.G.Bellemare, J.Menick, R.Munos, K.Kavukcuoglu	Google Deepmind	Lielbritānija	2017
4	Learning beyond simulated physics [4]	A.Asseman, T.Kornuta, A.Ozcan	IBM	ASV	2018

3.1.tabulas turpinājums

Nr.	Nosaukums	Autori	Iestāde	Valsts	Gads
5	Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks [26]	S.Bengio, O.Vinyals, N.Jaitly, N.Shazeer	Google	ASV	2018
6	Evaluating Current Machine Learning Techniques On Predicting Chaotic Systems [27]	S.Klinkachorn, J.Parmar	Stenfordas universitāte	ASV	2019
7	Deep learning of dynamics and signal-noise decomposition with time-stepping constraints [28]	S.H.Rudy, J.N.Kutz, S.L.Brunton	Vašingtonas Universitāte	ASV	2019
8	Machine learning for fast and reliable solution of time-dependent differential equations [29]	F.Regazzoni, L.Dede, A.Quarteroni	Milānas Politehniskā universitāte	Itālija	2019
9	Neural Ordinary Differential Equations [30]	T.Chen, Y.Rubanova, J.Bettencourt, D.Duvenaud	Toronto Universitāte	Kanāda	2019
10	Deep Learning with Real Data and the Chaotic Double Pendulum [2]	D.Gannon	Indianas Universitāte	ASV	2021
11	A simple equivariant machine learning method for dynamics based on scalars [31]	W.Yao, K.Fisher, David W, S. Villar	Ņujorkas Universitāte	ASV	2021
12	Deep neural network for system of ordinary differential equations: Vectorized algorithm and simulation [32]	T.T.Dufera	Adama Tehnoloģiju Universitāte	Etiopija	2021

3.2. tabulā apkopotas publikācijās aprakstītās sistēmas, kā arī apkopota informācija par citātu skaitu un izmantoto modeļu koda pieejamību. Pēc citātu skaita var spriest par publikācijas ietekmi uz nozari, bet koda pieejamība ļauj izpētīt praktisku publikācijā aprakstītās idejas implementāciju.

3.2. tabula.

Apskatīto publikāciju parametri.

Nr.	Nosaukums	Sistēma	Citēšanas reizes	Pieejams kods
1	Artificial Neural Networks for Solving Ordinary and Partial Differential Equations [1]	Ordinārie, parciāldiferenciālvienādojumi, to sistēmas	698	jā
2	Professor Forcing: A New Algorithm for Training Recurrent Networks [24]	Sequential MNIST datukopa	464	jā
3	Automated Curriculum Learning for Neural Networks [25]	Valodas modelēšana	368	jā
4	Learning beyond simulated physics [4]	Dubultais svārsts	9	jā
5	Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks [26]	MSCOCO attēlu datukopa	1568	jā
6	Evaluating Current Machine Learning Techniques On Predicting Chaotic Systems [27]	Dubultais svārsts	1	nē
7	Deep learning of dynamics and signal-noise decomposition with time-stepping constraints [28]	Nelineāra kubiskā atspere, Lorenca sistēma, zema Reinoldsa skaitļa plūsma ap cilindru, dubultais svārsts	116	jā
8	Machine learning for fast and reliable solution of time-dependent differential equations [29]	Vienkāršais svārsts ar slāpēšanu	67	nē
9	Neural Ordinary Differential Equations [30]	Ordināro diferenciālvienādojumu sistēmas	2029	jā
10	Deep Learning with Real Data and the Chaotic Double Pendulum [2]	Dubultais svārsts	0	nē
11	A simple equivariant machine learning method for dynamics based on scalars [31]	Atsperu dubultais svārsts	2	jā
12	Deep neural network for system of ordinary differential equations: Vectorized algorithm and simulation [32]	Ordināro diferenciālvienādojumu sistēmas	3	nē

Pēc publikāciju apkopojuma un analīzes, iespējams sniegt atbildes uz sākumā izvirzītajiem pētāmajiem jautājumiem:

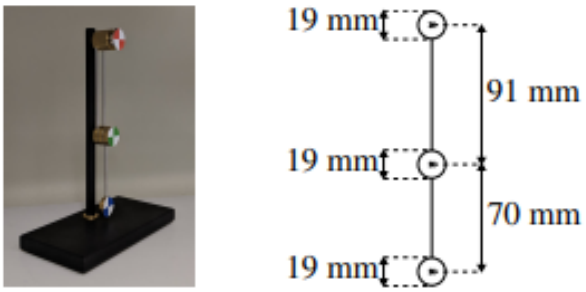
1. Kopumā no analizētajām 12 publikācijām 4 ietvēra eksperimentus ar dubultā svārsta sistēmas modelēšanu, izmantojot dziļajos neironu tīklos balstītas metodes, bet vēl divas dažādu citu tipa svārstus - parasto svārstu ar slāpēšanu un atsperu dubulto svārstu. Visbiežāk pielietotais modelēšanas elements ir LSTM šūnas vairākos slāņos, kurās padoti simulēti eksperimentu dati. Publikācijā Nr.6 veikts tiešs salīdzinājums starp LSTM un citām modelēšanas metodēm, un LSTM ieguva labākos rezultātus.

2. Apskatot publikācijās pielietotās metodes citu dinamisku sistēmu analizēšanai ar dziļo neironu tīklu metodēm, novērojamas divas pieejas - sistēmu aprakstošo fizikālu sakarību modelēšana, piem. modelējot vektoru lauka izmaiņas vai dažādus nezūdamības likumus. Kā arī sistēmu modelēšana izmantojot tikai eksperimentālus vai simulētus datus. Modelējot sistēmas ar to fizikālajām sakarībām pieminētas arī dažādas metodes diferenciālvienādojumu risināšanai.

3. Starp apskatītajām publikācijām vairākas izsaka priekšlikumus ilgtermiņa prognožu uzlabošanai. Kā piemēri apskatīti mācāmā uzdevuma sarežģītības palielināšana apmācības gaitā, kā arī metodes, kurās modelim tiek mācīts atšķirt prognozētās vērtības no reālajām, izmantojot ģeneratīvos sacīkstes tīklus GAN.

4. METODOLOĢIJA

Informācijas tehnoloģiju uzņēmuma IBM pētnieki 2019.gada jūlijā publicēja rakstu ”Learning beyond simulated physics” ar kuru kopā tika izdota datu kopa, kas satur datus par dubultā svārsta kinemātiku. Datu kopa satur video ar 21 eksperimentu ar dubultā svārsta svārstībām, katrs no tiem ir aptuveni 40s garš un satur ap 17500 kadru. Katrs video ir apstrādāts ar *OpenCV* programmatūru un no tiem iegūtas svārsta savienojumu vietu koordinātas katrā kadrā $x_1, y_1, x_2, y_2, x_3, y_3$. [4]



4.1. att. Eksperimentā lietotais svārsts un tā ģeometriskie parametri. [4]

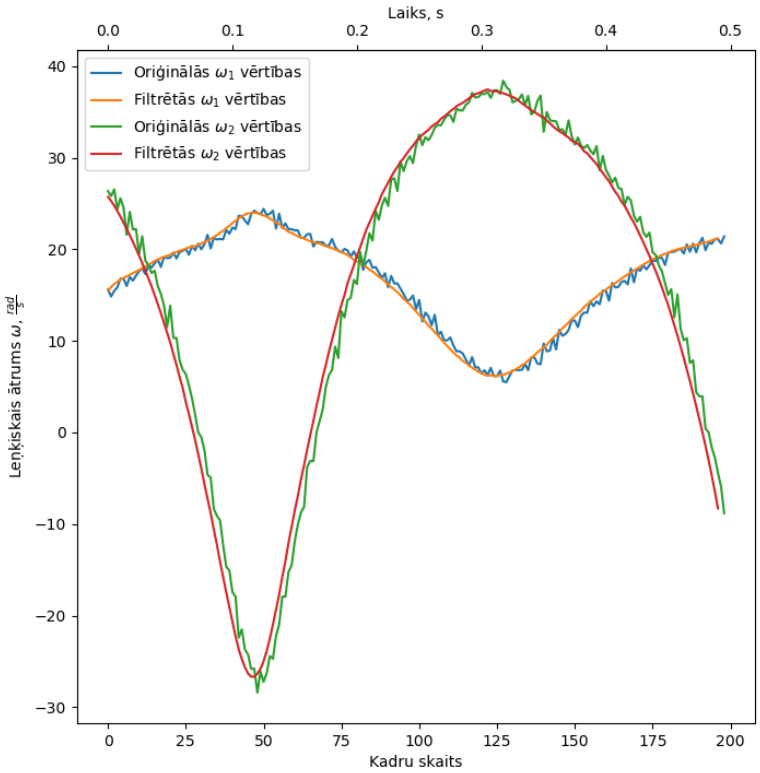
Svārstu aprakstošiem diferenciālvienādojumiem parametri ņemti pēc publikācijām norādītajām vērtībām, trūkstošie aproksimēti. Dziļo neironu tīklu ievades datu apstrāde apskatīta tālākajās nodaļās.

4.1. Datu apstrāde

Lai vienkāršotu sistēmas modelēšanu, savienojumu vietu koordinātas tika pārveidotas par leņķiem starp kustīgajām daļām θ_1 un θ_2. Turklāt lai dziļo neironu tīklu risinājumos nodrošinātu apmācības datu sēriju nepārtrauktību, tika izmantotas diferences starp šiem leņķiem secīgos kadros, praktiski iegūstot abu svārsta daļu leņķiskos ātrumus ω_1 un ω_2.

Ātrumu priekšrocība izpaužas iegūtā signāla nepārtrauktībā - nav novērojami leņķu vērtību lēcieni no 0 līdz 2π vai otrādi, kas notiktu kādai no svārsta daļām veicot pilnu apgriezienu ap savu asi. Tāpat arī pieņemt nepārtrauktas leņķu vērtības nebūtu izdevīgs risinājums, jo tās būtu atkarīgas ne tikai no konkrētās svārsta daļas atrašanās vietas, bet arī no iepriekš veikto pilno apgriezienu skaita.

Tā kā kadru uzņemšanas ātrums ir galīgs (400 kadri/s), iegūtais signāls satur troksni. Tā filtrēšanai tiek izmantots Savitska - Golā digitālais filtrs, kura praktiskā implementācija veikta *Python* bibliotēkā *ScyPy*.[33] Rezultātu salīdzinājums ar oriģinālo signālu redzams 4.2.att.:



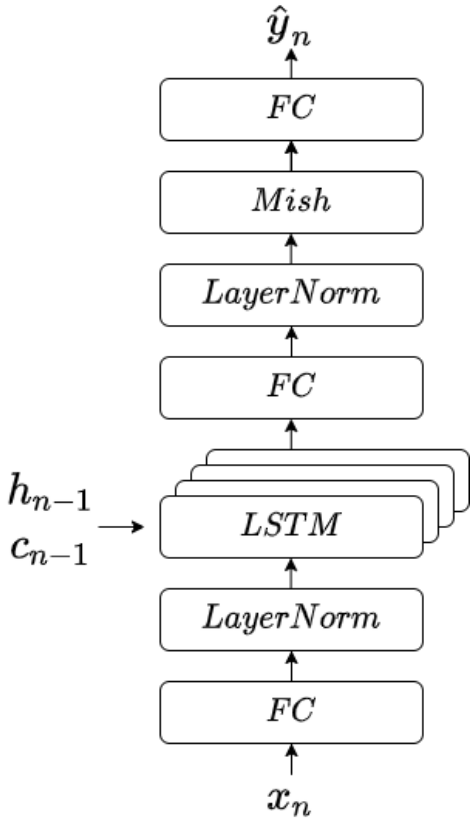
4.2. att. Trokšņainu un filtrētu signālu salīdzinājums.

Dziļo neironu tīklu risinājumos kā ievades un izvades dati izmantoti leņķiskie ātrumi ar sērijas garumu 400, ģenerēti ar slīdošu logu ejot pāri katrai no pilnajām eksperimentu datu sērijām ar soļa garumu 1. Rezultātā no pilnās datu kopas iegūti gandrīz 400 tūkstoši datu sēriju. Lai objektīvi novērtētu modeļa veiktspēju, datu kopa tiek sadalīta apmācības un testa daļās ar attiecību 80:20. Tātad 80% datu tiek izmantoti modeļa apmācības procesā, bet ar atlikušajiem 20% tiek tikai pārbaudītas kļūdas vērtības, nekoriģējot modeļa parametrus. Šādi var pārbaudīt situāciju, kad modelis precīzi iemācījies ievades datus un dod šķietami labas kļūdas vērtības, bet nespēj dot labu prognozi vēl neredzētām vērtībām. Šādu parādību sauc par pārmācīšanos (*overfitting*) un tā ir fundamentāla problēma dziļo neironu tīklu apmācības procesā.

Kā papildus signāla apstrāde pielietota signāla retināšana - signāla datu punktu skaita samazināšana, jauno signālu veidojot no vecā signāla katra n-tā datu punkta. Piem. signāla retināšana ar pakāpi 3 veidos jaunu signālu, izmantojot vecā signāla katru trešo vērtību, rezultātā signāls aptvers to pašu novērojumu periodu, bet datu punktu skaits būs mazāks. Šāda veida apstrāde noder, lai samazinātu datus saturošo matricu izmērus un līdz ar to arī mākslīgo neironu tīklu apmācības ilgumu. Tomēr tās noderīgums jātestē, jo datu apjoma zaudēšana var samazināt modeļa galējo veiktspēju.

4.2. LSTM risinājums

Modelēšanai ar dziļajiem neironu tīkliem kā pamatelementi izmantotas LSTM šūnas vairākos slāņos. Veidojot modeļa arhitektūru tika ņemti vērā arī pētījumi, kas liecina par rekurento modeļu veiktspējas uzlabošanos un ātrāku konverģenci, ja ievades vērtības tiek laistas cauri lineāram slānim. Tāpat pēc LSTM šūnām izmantots daudzslāņu vienvirziena tīkls ar *Mish* aktivācijas funkciju un *LayerNorm* normalizācijas slāni. [34] [35] [36] Lai novērstu pārlieku lielu kļūdas vērtību pirmajā modeļa darbības iterācijā, tiek izmantoti apmācāmi sākotnējie stāvokļa un slēptā stāvokļa vektori c_0 un h_0. Kopējā modeļa viena soļa shēma redzama attēlā:



4.3. att. Dziļo neironu tīklu modeļa viena soļa shēma.

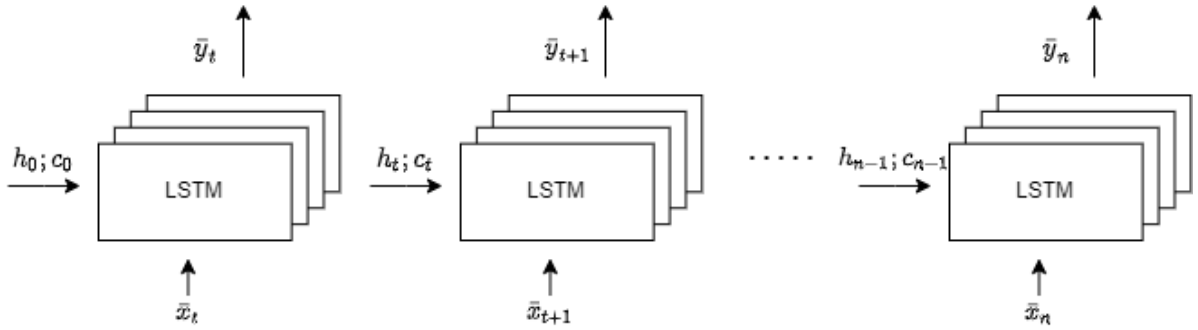
Kā kļūdas funkcija izmantota vidējā absolūtā novirze *(MAE - mean absolute error)*. Šāda kļūdas funkcija tiek pielietota, lai saglabātu lineāru sakarību starp kļūdas vērtību un leņķiskā ātrumā mērvienību - rad/s, tādā veidā piešķirot validācijas procesam saprotamākas vērtības.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid y_i - \hat{y}_i \mid \tag{4.1}$$

kur: y – patiesā eksperimentālā vērtība (*ground truth*)
$\hat{y}$ – modeļa prognozētā vērtība

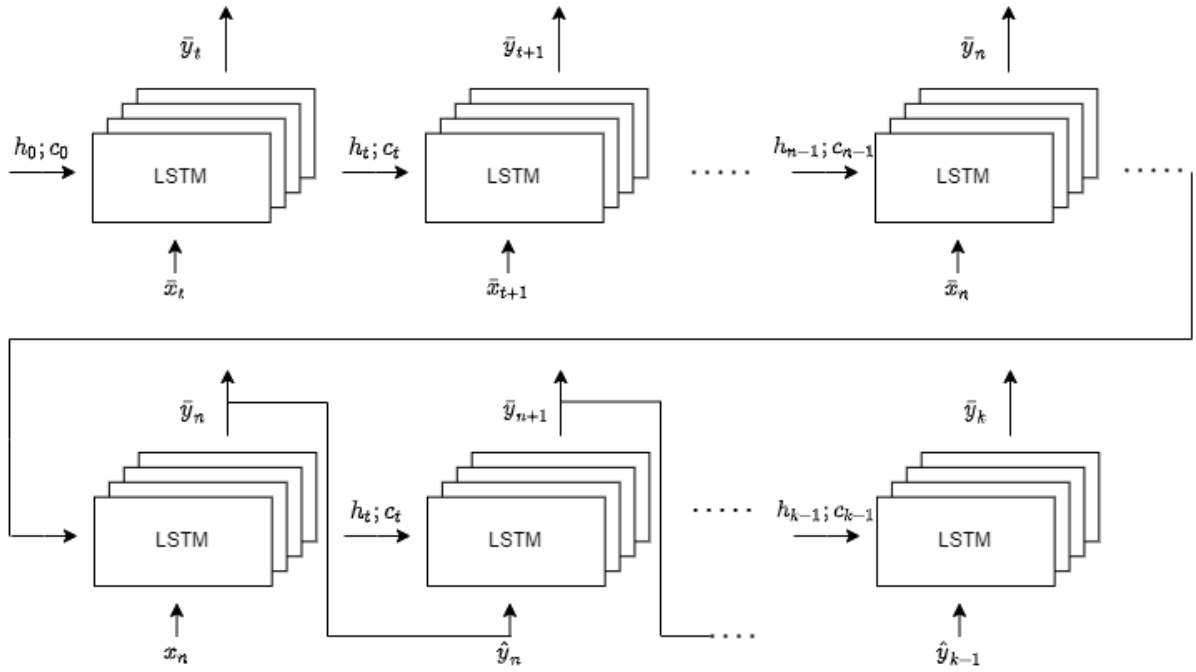
Optimizācijai pielietots *Adam* algoritms (*adaptive moment estimation*), kurš izmanto gradienta metodi tāpat kā kalnākāpēja algoritms, taču papildus ļauj izmantot apmācības soļa adaptīvu koriģēšanu apmācības gaitā balstoties uz statistiskām metodēm.[37] Pētījumi rāda, ka konkrētais algoritms nodrošina labus rezultātus dziļo neironu tīklu uzdevumos salīdzinot ar citiem algoritmiem, tāpēc arī tas tiks pielietots darbā apskatītā uzdevuma modelēšanai.[38]

Lai modeli apmācītu, ievades sērijas dati tiek laisti cauri attiecīga izmēra shēmai. Modeļa arhitektūras sērijas attēlojums parādīts attēlā:



4.4. att. Dziļo neironu tīklu sērijas modeļa shēma.

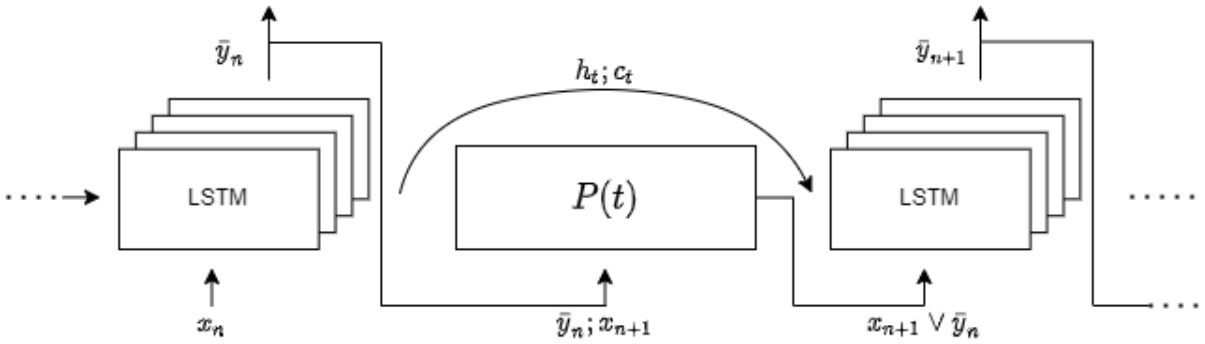
Šādā izkārtojumā modelis katrā laika solī prognozē nākamā soļa vērtību, tomēr tas nodrošina apmācību tikai viena soļa intervālā. Lai veicinātu modeļa ilgtermiņa prognozēšanas spējas, papildus viena soļa prognozēm tiek izmantots rekurents process, kura laikā modeļa ievades vērtība kādā laika solī ir nevis vērtība no datu kopas x_n, bet gan iepriekšējā soļa paša modeļa prognoze.



4.5. att. Dziļo māsklīgo neironu tīklu modeļa apmācības shēma ar viena soļa prognozes daļu un rekurento ilgtermiņa prognozes daļu.

Šādā apmācības konfigurācijā modelis pirmos n soļus tiek apmācīts viena soļa režīmā, bet pēc noteiktās robežas sasniegšanas pārslēdzas uz rekurento apmācības režīmu. Tādā veidā modelis iemācās veikt gan precīzu nākamā soļa prognozi, gan spēt turpināt prognozi vairāku soļu intervālā.

Tāpat pētījumi liecina, ka ilgtermiņa prognožu uzlabošanai pielietojama pakāpeniskas sarežģītības tehnika. Tā balstās idejā, ka mākslīgo neironu tīkla apmācība jāsāk ar vienkāršākajām sakarībām un apmācības gaitā pakāpeniski jāpaaugstina ievades datu sarežģītība.[39] Praktiski šāda tehnika implementēta rekurentajā apmācības daļā, kur apmācības plāns nosaka varbūtību, vai nākamā ievades vērtība tiks padota kā vērtība no datu kopas, vai nu iepriekšējā soļa modeļa prognoze.



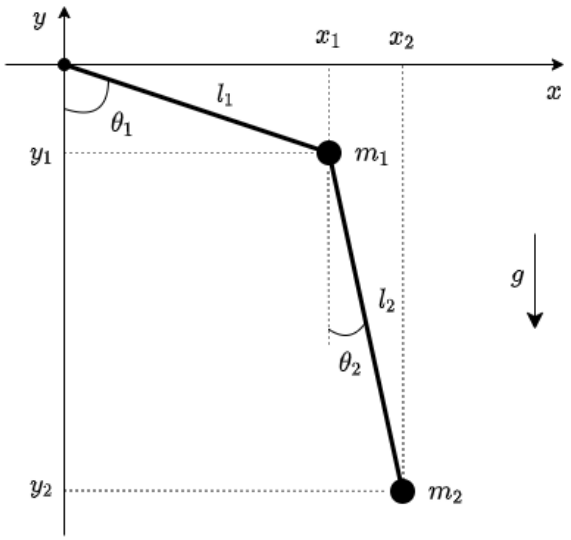
4.6. att. Dziļo neironu tīklu modeļa apmācības shēma ar pakāpeniskas sarežģītības tehniku.

Apmācības gaitā varbūtība padot modeļa iepriekš ģenerēto vērtību palielinās, tādā veidā nodrošinot pakāpenisku pāreju no datu kopas ievades datiem uz ģenerētajiem datiem. Šādas metodes ieguvums ir sākotnējās kļūdas samazināšana, kas savukārt apmācības procesu padara stabilāku un potenciāli ļauj izmantot garāku rekurento apmācības daļu, veicinot ilgtermiņa prognozes uzlabošanos.

4.3. Matemātiskais modelis

Jau iepriekš pieminēto eksperimentu video redzams, ka apskatāmā dubultā svārsta sistēma būtībā ir trīs brīvības pakāpju sistēma - papildus abu svārsta daļu kustībai novērojamas arī paša svārsta pamatnes svārstības ar zemu amplitūdu.

Vienkāršošanai apskatāmā dubultā svārsta sistēma tiks modelēta kā divu brīvības pakāpju sistēma, to ērti aprakstīt ar Lagranža vienādojumiem. Tāpat tiek pieņemts, ka kustība notiek starp divām lodītēm ar masām m_1 un m_2, kuras savienotas ar cietiem savienojumiem ar garumiem l_1 un l_2. Uz abām lodītēm darbojas gravitācijas un gaisa pretestības spēks. 4.7.att. parādīta izmantotā modeļa shēma.



4.7. att. Dubultā svārsta modelēšanas shēma

Lagranža vienādojums:

$$\frac{d}{dt}(\frac{\partial K}{\partial \dot{q}_i}) - \frac{\partial K}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} + \frac{\partial P}{\partial q_i} = Q_i \tag{4.2}$$

kur: K – kinētiskā enerģija
P – potenciālā enerģija
D – disipatīvā enerģija
Q – spēks
q – vispārināta koordināta

Koordinātas pirmajai svārsta daļai:

$$x_1 = l_1 \sin\theta_1 \qquad\qquad y_1 = -l_1 \cos\theta_1 \tag{4.3}$$

Koordinātas otrajai svārsta daļai:

$$x_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2 \qquad y_2 = -l_1 \cos\theta_1 - l_2 \cos\theta_2 \tag{4.4}$$

Ātrumi, koordinātas atvasinot pēc laika:

$$\dot{x}_1 = l_1\dot{\theta}_1 \cos\theta_1 \qquad \dot{y}_1 = l_1\dot{\theta}_1 \sin\theta_1 \tag{4.5}$$

$$\dot{x}_2 = l_1\dot{\theta}_1 \cos\theta_1 + l_2\dot{\theta}_2 \cos\theta_2 \qquad \dot{y}_2 = l_1\dot{\theta}_1 \sin\theta_1 + l_2\dot{\theta}_2 \sin\theta_2 \tag{4.6}$$

Potenciālā enerģija:

$$\begin{aligned} P &= m_1 g y_1 + m_2 g y_2 \\ &= -(m_1 + m_2) l_1 g \cos\theta_1 - m_2 l_2 g \cos\theta_2 \end{aligned} \tag{4.7}$$

Kinētiskā enerģija:

$$\begin{aligned} K &= \frac{m_1 v_1^2}{2} + \frac{m_2 v_2^2}{2} \\ &= \frac{m_1(\dot{x}_1^2 + \dot{y}_1^2)}{2} + \frac{m_2(\dot{x}_2^2 + \dot{y}_2^2)}{2} \\ &= \frac{m_1 l_1^2 \dot{\theta}_1^2}{2} + \frac{m_2(l_1^2\theta_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2))}{2} \end{aligned} \tag{4.8}$$

Disipatīvā enerģija:

$$\begin{aligned} D &= \frac{c_1 v_1^2}{2} + \frac{c_2 v_2^2}{2} \\ &= \frac{c_1(\dot{x}_1^2 + \dot{y}_1^2)}{2} + \frac{c_2(\dot{x}_2^2 + \dot{y}_2^2)}{2} \\ &= \frac{c_1 l_1^2 \dot{\theta}_1^2}{2} + \frac{c_2(l_1^2\theta_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2))}{2} \end{aligned} \tag{4.9}$$

Atvasinājumi:

$$\begin{aligned}\frac{\partial K}{\partial \dot{\theta}_1} &= m_1 l_1^2 \dot{\theta}_1 + m_2 l_1^2 \dot{\theta}_1 + m_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \\ &= (m_1 + m_2) l_1^2 \dot{\theta}_1 + m_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_1 - \theta_2)\end{aligned} \tag{4.10}$$

$$\frac{d}{dt}(\frac{\partial K}{\partial \dot{q}_i}) = \ddot{\theta}_1 l_1^2 (m_1 + m_2) + m_2 l_1 l_2 \ddot{\theta}_2 cos(\theta_1 - \theta_2) - m_2 l_1 l_2 sin(\theta_1 - \theta_2)(\dot{\theta}_1 - \dot{\theta}_2) \tag{4.11}$$

$$\frac{\partial K}{\partial \theta_1} = -m_2 l_1 l_2 sin(\theta_1 - \theta_2)\dot{\theta}_1 \dot{\theta}_2 \tag{4.12}$$

$$\frac{\partial P}{\partial \theta_1} = (m_1 + m_2) g l_1 sin(\theta_1) \tag{4.13}$$

$$\frac{\partial D}{\partial \dot{\theta}_1} = (c_1 + c_2) l_1^2 \dot{\theta}_1 + c_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \tag{4.14}$$

Ievietojot visus atvasinājumus Lagranža vienādojumos un saīsinot iegūst diferenciālvienādojumu pirmajai brīvības pakāpei:

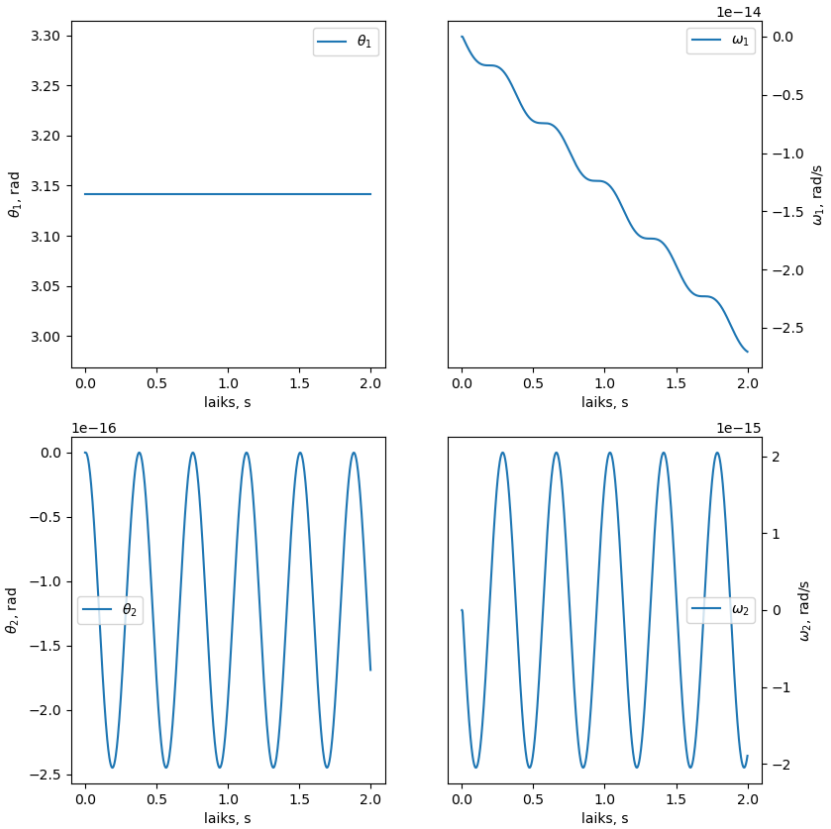
$$\begin{aligned}&(m_1 + m_2) l_1 \ddot{\theta}_1 + m_2 l_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) + m_2 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) \\ &+(m_1 + m_2) g \sin\theta_1 + (c_1 + c_2) l_1^2 \dot{\theta}_1 + c_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_1 - \theta_2) = 0\end{aligned} \tag{4.15}$$

Līdzīgi atvasinot enerģiju vienādojumus pēc θ_2 un savelkot iegūst otru diferenciālvienādojumu:

$$\begin{aligned}m_2 l_2 \ddot{\theta}_2 + m_2 l_1 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) - m_2 l_1 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + m_2 g \sin\theta_2 \\ +c_2 l_2 \dot{\theta}_2 + c_2 l_1 l_2 cos(\theta_1 - \theta_2)\dot{\theta}_1 = 0\end{aligned} \tag{4.16}$$

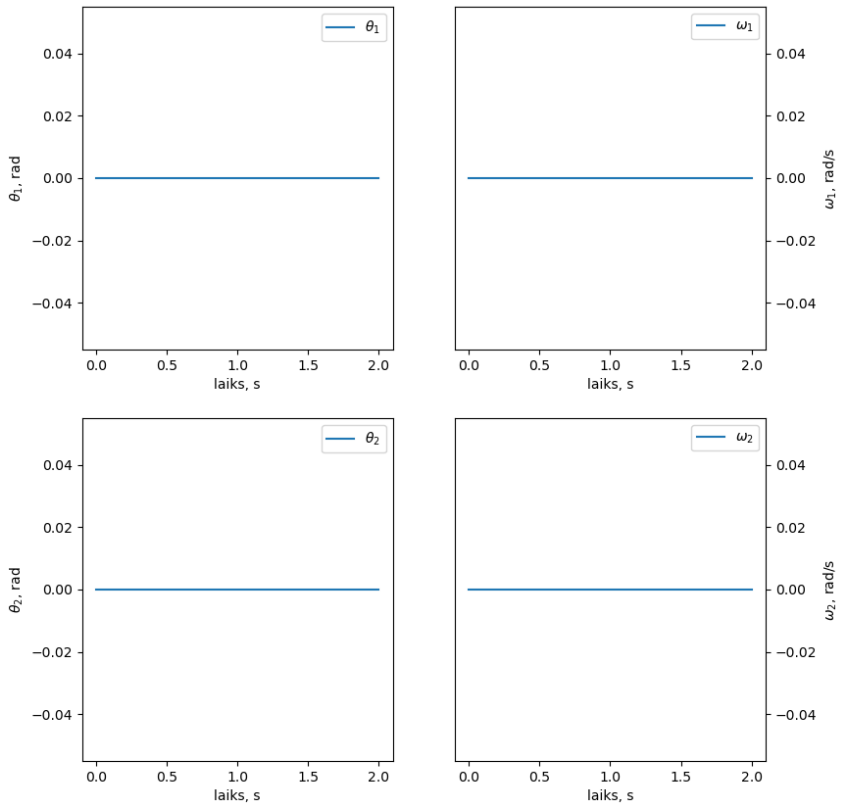
Diferenciālvienādojuma atrisināšanai tiek izmantota *Python* bibliotēkas *SciPy* iebūvēta funkcija *solve ivp*, kas izmanto RK45 algoritmu diferenciālvienādojumu atrisināšanai. [40]

Lai pārliecinātos par diferenciālvienādojuma korektumu, iespējams pārbaudīt svārsta parametrus līdzsvara stāvokļos - stabilais līdzsvara stāvoklis $y_0 = \{0, 0, 0, 0\}$, nestabilais līdzsvara stāvoklis $y_0 = \{\pi, 0, 0, 0\}$ attiecīgi formātā $\{\theta_1, \omega_1, \theta_2, \omega_2\}$. Korekta diferenciālvienādojuma sastādīšanas gadījumā ievadot līdzsvara stāvokļa parametrus iegūtajiem parametriem jābūt konstantiem.



4.8. att. Nestabila līdzsvara stāvokļa diferenciālvienādojuma rezultāti.

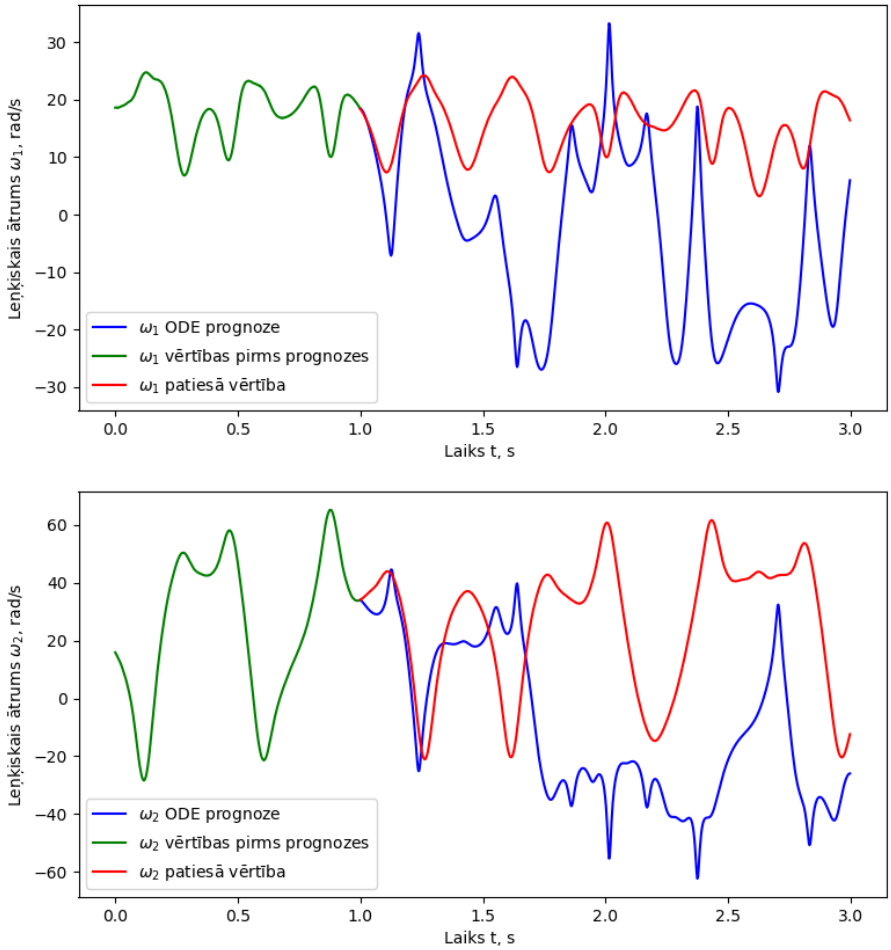
Kā redzams attēlā, nestabila līdzsvara stāvokļa parametri nesaglabājas konstanti, taču to izraisa aprēķinu galīgā precizitāte. Galīgs precizitātes gadījumā $sin(\pi) \approx 10^{-16}$, nevis precīzi 0, tāpēc kļūdai sakrājoties novērojamas nobīdes. Šajā gadījumā novērojamās nobīdes arī atbilst šāda mēroga kļūdai. Speciāli definējot, ka ļoti mazas vērtības tiek noapaļotas uz 0, svārstības nav novērojamas.



4.9. att. Stabila līdzsvara stāvokļa diferenciālvienādojuma rezultāti.

Modelējot stabilo līdzsvara gadījumu līdzīgas nobīdes nav novērojamas, jo arī ar galīgu skaitļošanas precizitāti *sin(0) = 0*. Tātad var secināt, ka diferenciālvienādojums sastādīts korekti.

Kad iegūts apstiprinājums par diferenciālvienādojumu korektumu, tiek veikta modelēšana, sākuma nosacījumus iegūstot no eksperimentālajiem datiem un veicot salīdzinājumu starp diferenciālvienādojuma prognozi un eksperimentālajiem datiem.



4.10. att. Diferenciālvienādojuma un eksperimentālo rezultātu salīdzinājums leņķiskajiem ātrumiem ω_1 un ω_2.

Kā redzams no 4.10.att. salīdzinājuma starp diferenciālvienādojuma iegūtajām vērtībām un eksperimentālajām vērtībām, diferenciālvienādojuma vērtības atbilst eksperimentālajām tikai pirmajos prognozes soļos, pēc tam novērojamās vērtības diverģē. Skaitlisks novērtējums salīdzinājumam apskatīts nodaļā pie rezultātiem.

4.4. Apmācību un validācijas protokols

Modeļu prognozēšanas spēju kvantitatīvai salīdzināšanai ieviestas metrikas, kuras raksturos prognozes no vairākiem aspektiem. Pārbaudei vienmērīgi no katra eksperimenta atlasītas datu sērijas, kopumā 128, no kurām katra atbilst 5 sekunžu svārstībām. No tām pirmā sekunde padota modelī kā ievades vērtības, bet pārējās salīdzinātas ar modeļa prognozi, tā novērtējot modeļa spējas. Arī ievades dati visām turpmākajos attēlos redzamajām shēmām ņemti no vienām un tām pašām sērijām, līdz ar to var veikt tiešu salīdzinājumu.

Jāņem vērā, ka validācijas sēriju garuma atšķirību no testa kopas dēļ nav iespējams

nodrošināt, ka modelis tās nav redzējis apmācības procesā, bet identiski parametri pielietoti visu eksperimentu validācijai, tāpēc jebkura potenciālā nobīde ir vienāda visiem modeļiem un neietekmē savstarpējo salīdzinājumu.

Konkrēti tiek pielietotas divas metrikas. Pirmā novērtē modeļa prognozēšanas spējas 2s intervāla prognozēšanai - modeļa prognoze katrā solī tiek salīdzināta ar reālajiem datiem un aprēķināta starpība, galējā metrika ir visu iegūto starpību vidējā vērtība. Katrā solī kā ievades vērtība tiek pielietota iepriekšējā solī prognozētā vērtība.

$$MAE = \frac{1}{n(t)} \sum_{i=1}^{t=2s} \mid y_i - \hat{y_i} \mid \tag{4.17}$$

$$x_i = \hat{y}_{i-1} \tag{4.18}$$

kur: $n(t)$ – datu punktu skaits t sekundēs
y_i – patiesā eksperimentālā vērtība konkrētajā solī
$\hat{y}_i$ – modeļa prognozētā vērtība konkrētajā solī
$\hat{y}_{i-1}$ – modeļa prognozētā vērtība iepriekšējā solī
x_i – modeļa ievades vērtība konkrētajā solī

Papildu prognožu novērtēšanai ieviesta otra metrika. Tā veic prognožu salīdzināšanu ar patiesajā vērtībām tāpat kā iepriekš aprakstītā metrika, bet atšķirībā no tās fiksē prognozēto soļu skaitu, līdz kādā solī kļūdas vērtība pārsniedz parametrā noteiktu vērtību, šajā gadījumā 8 rad/s.

$$MS = i, \text{ kad } \mid y_i - \hat{y_i} \mid > \delta \tag{4.19}$$

kur: δ – kļūdas maksimuma parametrs
i – konkrētā soļa numurs

Turpmākajā tekstā tā tiks dēvēta kā ilgtermiņa MS kļūda. Jāņem vērā, ka modeļiem ar dažādām signāla retināšanas pakāpēm vienāds soļu skaits apskata dažādus laika periodus.

$$MS_{norm} = R \cdot MS \tag{4.20}$$

kur: R – signāla retināšanas pakāpe

Objektivitātes saglabāšanai MS vērtības tiek normalizētas pēc signāla retināšanas pakāpes, lai dažādu modeļu MS vērtības reprezentētu vienādu laika vērtību katram solim.

Tāpat apmācības procesā dziļo mākslīgo neironu tīklu modeļos ir daudzas maināmas iestatījumu vērtības, tāpēc pieņemts apmācīt daudzus modeļus, veicot parametru režģa pārmek-

lēšanu, un pēc tam salīdzināt to veiktspēju pēc noteiktiem validācijas kritērijiem. Konkrētā dziļo neironu tīklu modeļa apmācība veikta ar RTU HPC centra piedāvātajām K40 un V100 videokartēm. Konkrētajā modelī pielietotās apmācību definējošas iestatījumu vērtības:

- datu partijas lielums - 180
- slēptā slāņa izmērs - 256
- LSTM slāņu skaits - 4
- Apmācību ātrums - 10^{-3}

Pie izmantotās konfigurācijas dziļā mākslīgo neironu tīkla koriģējamo parametru skaits ir 2 542 082. Iepriekš minētie iestatījumi apmācību laikā netiek mainīti, bet ar pārējiem tiek veikta parametru režģa pārmeklēšana, lai pārbaudītu modeļu veiktspēju ar dažādām savstarpējām iestatījumu kombinācijām:

- Vai pielietota signāla retināšana.
- Signāla retināšanas pakāpe.
- Vai pielietota rekurentā daļa kļūdas aprēķinā.
- Rekurentās daļas garums.
- Vai pielietota pakāpeniska grūtība.

Praktiska minēto dziļo mākslīgo neironu tīklu metožu implementācija veikta ar *PyTorch* bibliotēkas palīdzību programmēšanas vidē *Python*. Tā ir atvērtā koda implementācija, pieejama publiskā *GitHub* saitē `https://github.com/reinisfreibergs/bakalaura_darbs.git`

REZULTĀTI

Eksperimentos iegūtas MAE un MS kļūdu vērtības apkopotas tabulā. Tās sarindotās secībā pēc labākajām MS kļūdu vērtībām, kuras normalizētas pēc signāla retināšanas reizēm. Kā metodes atzīmetas ODE un LSTM, attiecīgi diferenciālvienādojumu risinājumi un dziļo mākslīgo neironu tīklu risinājumi.Tāpat tabulā apkopota informācija par izvēles iestatījumiem - kādā pakāpē un vai pielietota signāla retināšana, vai LSTM apmācības laikā ieslēgta rekurentā daļa kļūdas aprēķinā un tās garums, vai izmantota pakāpeniska grūtība LSTM apmācībai.

5.1. tabula.

Rezultātu salīdzinājums

Metode	Signāla retināšanas reizes	Rekurentā daļa kļ. aprēķinā	Kļūdas funkc. rek. daļas garums, soļi	Pakapēniska grūtība	MS kļūda, soļi	MAE kļūda, rad/s
LSTM	1	+	50	+	600	0.0126
LSTM	2	+	50	+	504	0.0153
LSTM	1	+	50	-	444	0.0147
LSTM	2	+	25	+	344	0.0189
LSTM	4	+	12	+	296	0.0215
LSTM	4	+	50	+	248	0.0233
LSTM	1	-	-	-	190	0.0262
LSTM	2	+	25	-	143	0.0198
LSTM	4	+	50	-	74	0.0219
LSTM	2	+	50	-	72	0.0250
LSTM	4	+	12	-	45	0.0269
ODE	1	-	-	-	30	0.0392
ODE	2	-	-	-	24	0.0390
ODE	4	-	-	-	13	0.0422

Kā redzams 5.1. tabulā, visu LSTM modeļu prognozes ir labākas par diferenciālvienādojumu prognozēm. Lai objektīvi pārbaudītu izteikumu, tiks veiktas statistiskās pārbaudes un noteikts, vai LSTM pieeja tiešām dod labākus rezultātus. Konkrētajā pārbaudē noteikts, vai labākās LSTM metodes MAE kļūdas vērtība ar statistisku nozīmīgumu ir mazāka par labākās ODE metodes MAE kļūdas vērtību.

Tiek definētas hipotēzes:

$$h_0 : \bar{x}_A = \bar{x}_B \tag{4.21}$$

$$h_a : \bar{x}_A < \bar{x}_B \tag{4.22}$$

Pieņemot, ka abu metožu kļūdu vērtības atbilst normālajam sadalījumam un zinot, ka mērījumu apjoms ir liels (128), salīdzināšanai izmantots normālā z sadalījuma formulas.
Pie ticamības 0.99 var nolasīt kritisko z vērtību:

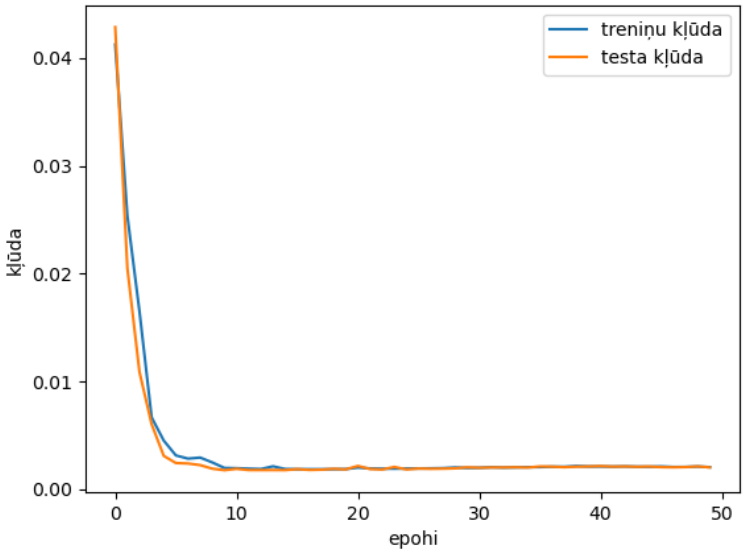
$$z_\alpha = -2.575 \tag{4.23}$$

Eksperimentālā z vērtība:

$$z_{exp} = \frac{\bar{x}_A - \bar{x}_B}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} = \frac{0.012613 - 0.039210}{\sqrt{\frac{0.0095^2}{128} + \frac{0.01245^2}{128}}} = -19.21 \tag{4.24}$$

Vērtība -19.21 ir mazāka par kritisko -2.575, tātad nulto hipotēzi var noraidīt. Tātad ar 99% pārliecību var teikt, ka LSTM modeļa MAE kļūdas vērtība ir mazāka par ODE modeļa MAE kļūdas vērtību.

Tāpat modeļu apmācības procesu var novērtēt, salīdzinot apmācības un testa daļas kļūdu vērtības katrā epohā, jo to manāma atšķirība norādītu uz pārmācīšanos.



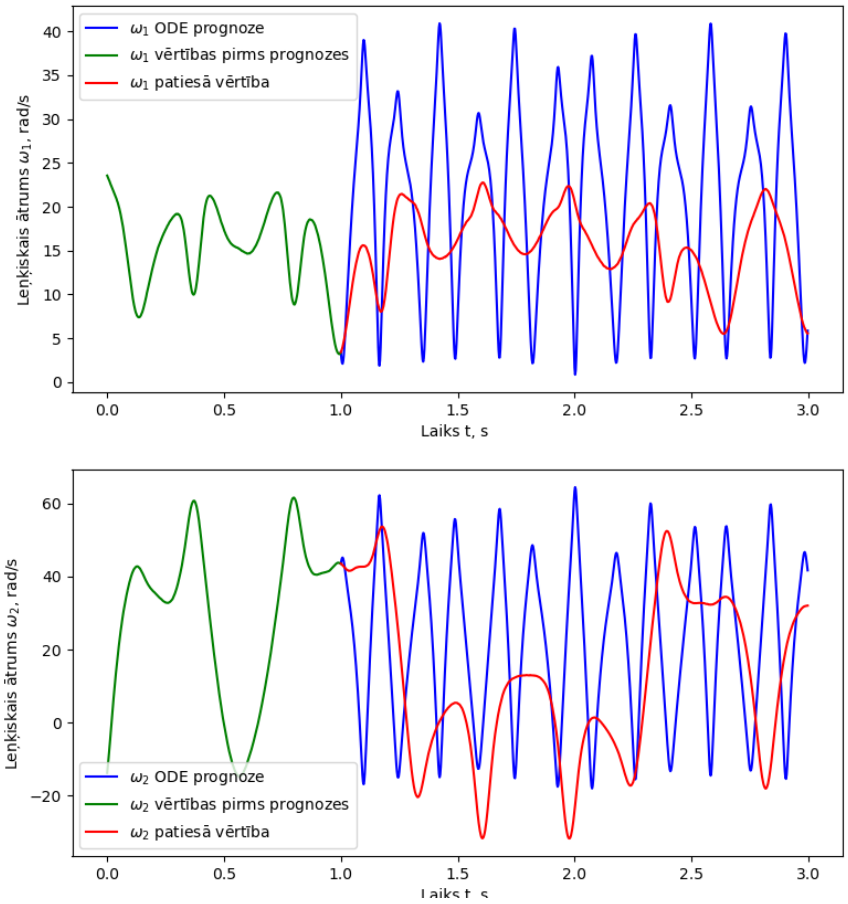
5.1. att. Apmācības un testa daļas kļūdu vērtības atkarībā no epoha.

Kā redzams attēlā, apmācību un testa kļūdu vērtību līknes konverģē, tātad modeļa parametri izvēlēti korekti. Tomēr šāda pārbaude ir noderīga, jo abu līkņu manāmu atšķirību

gadījums liecinātu par pārmācīšanos - modelis būtu spējis iemācīties tikai kādas konkrētas ievades datu īpašības, bet nespēj veikt jēdzīgas prognozes ārpus tiem.

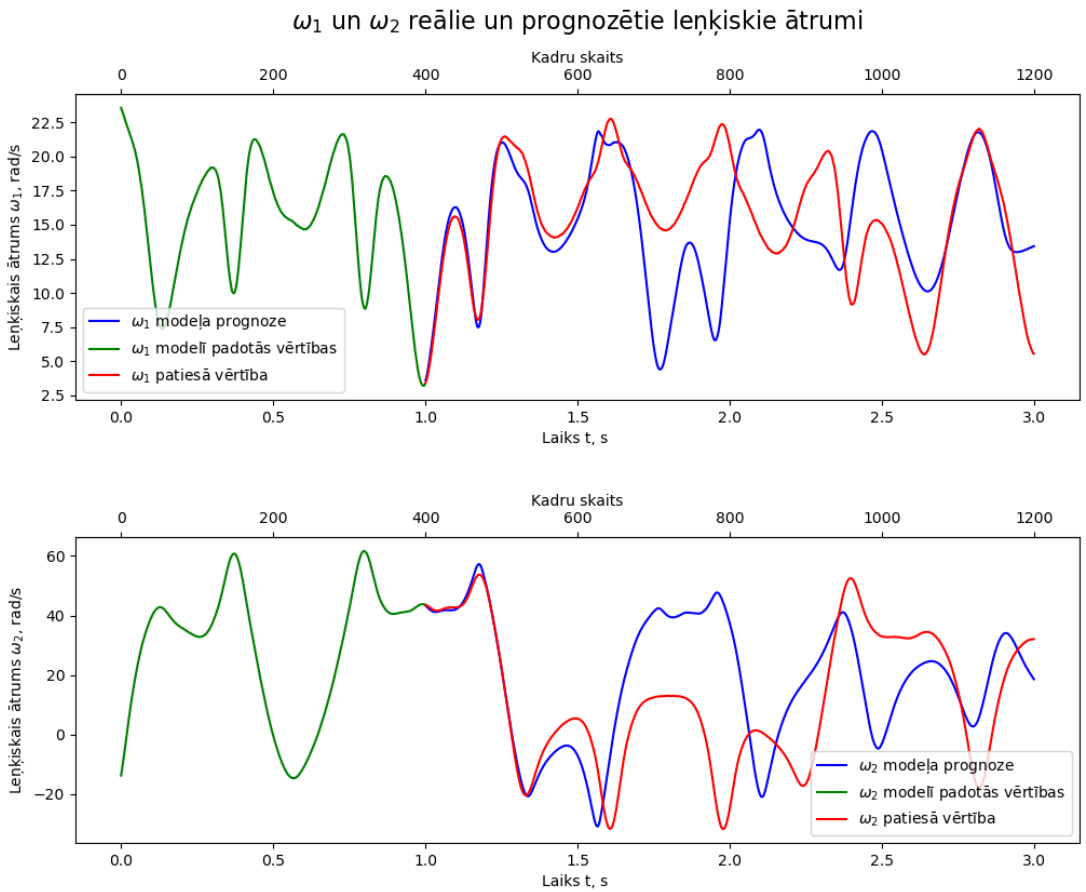
Labākajam modelim apmācības ilgums ir aptuveni 33h. Ņemot vērā ilgo apmācības laiku, visi modeļi apmācīti 50 epohus, nepārbaudot pilnīgu kļūdas funkcijas konverģenci uz minimālo vērtību. Tomēr rezultātu kļūdu vērtību līknes liecina, ka konverģence sasniegta lielākajai daļai modeļu.

Papildus metrikām modeļa spējas var novērtēt arī vizuāli, salīdzinot ODE un LSTM metožu prognozes ar reālajām vērtībām. Tā var novērtēt gan metožu spējas sekot precīzi reālajām vērtībām prognozēšanas sākumposmā, gan arī kopējas prognozes tendences. 5.2.att. salīdzinātas diferenciālvienādojuma prognozes ar eksperimentālajiem datiem.



5.2. att. Diferenciālvienādojuma un eksperimentālo rezultātu salīdzinājums leņķiskajiem ātrumiem ω_1 un ω_2.

Savukārt 5.3.att. salīdzinātas dziļo mākslīgo neironu tīklu prognozes ar eksperimentālajiem datiem. Jāņem vērā, ka 5.2.att. un 5.3.att. redzamie salīdzinājumi veikti pie vienādiem ievades datiem, tāpēc tos var izmantot kā tiešu mērauklu.



5.3. att. Dziļo neironu tīklu un eksperimentālo rezultātu salīdzinājums leņķiskajiem ātrumiem ω_1 un ω_2.

Kā redzams augstāk parādītajos attēlos, arī šeit LSTM modeļa prognozēšanas spējas ir vizuāli labākas - tās spēj ilgāk veikt precīzu prognozi tās sākumā, kā arī ilgtermiņā saglabā līdzīgas tendences reālajam signālam. ODE prognozes ir spējīgas saglabāt leņķiskā ātruma izmaiņas virzienus, tomēr gan prognozes sākumā, gan arī ilgtermiņā ir salīdzinoši vājas.

TĀLĀKIE PĒTĪJUMI

Darbā apskatītais dziļo neironu tīklu risinājums balstās uz LSTM šūnām un kā ievades datus izmanto tikai eksperimentālās svārsta posmu leņķisko ātrumu vērtības. Kā alternatīva šāda tipa modelim ir jau literatūras apskatā pieminētās sistēmas, kuras papildus eksperimentālajiem datiem pielieto arī par sistēmu zināmās fizikālās sakarības. Tādā gadījumā kļūdas funkcija sastāv no divām daļām - tiek optimizēta gan modeļa prognožu līdzība eksperimentālajiem datiem, gan arī vai tās atbilst zināmajām fizikālajām sakarībām. Šādu modeļu priekšrocība potenciāli varētu būt labāka ilgtermiņa prognozēšanas spēja, jo tā piešķir papildu informāciju, kuru dziļie neironu tīkli spēj pielietot. Turklāt ar koeficientiem iespējams koriģēt abu kļūdas funkcijas daļu īpatsvaru, tādā veidā prioritizējot kādu no tām. Tāpat šādi modeļi varētu sniegt jēdzīgas prognozes ar zemāku eksperimentālo datu apjomu, jo zināšanas par fizikālajām sakarībām dotu papildus informāciju par sistēmu.

Tāpat apskatīto risinājumu varētu pielietot cita veida sistēmām, kuras dotu arī praktisku pielietojumu. Tā kā dziļajos māksīgo neironu tīklos balstītais modelis uzrādīja labas prognozēšanas spējas, to varētu pielietot kādas praktiskas sistēmas diagnosticēšanai. Ņemot vērā tīklu spēju apstrādāt lielu apjomu datu, tas varētu dot priekšrocības tādu sistēmu modelēšanai, kuras darbojas mainīgos apstākļos un kur sarežģīti noteikt rezultātu ar standarta metodēm.

Pētījumus varētu arī turpināt, lai vēl vairāk uzlabotu jau apskatītā modeļa prognozēšanas spējas. Tālāka izpēte par alternatīvām LSTM šūnai, kas potenciāli dotu citu šūnu ar labākām atmiņas spējām, kā arī padziļinātāka izpēte tehnikām, kas ļauj uzlabot ilgtermiņa prognožu spējas.

SECINĀJUMI

Modelējot dubultā svārsta sistēmu un salīdzinot diferenciālvienādojuma rezultātus ar dziļo mākslīgo neironu tīklu metožu rezultātiem, mākslīgo neironu tīklu metodes sasniedz labākus rezultātus gan pēc ieviestajām MAE un MS novērtēšanas vērtībām, gan arī pēc vizuālas grafiku novērtēšanas. Labākais dziļo mākslīgo neironu tīklu modelis spēj uzrādīt MAE kļūdu 0.0126 rad/s vērtībā, turpretim diferenciālvienādojuma labākā MAE kļūda ir 0.0422 rad/s. Līdzīgi labākā MS kļūda LSTM modelim ir 600 soļi jeb 1.5s prognozes intervāla, bet labākā prognoze diferenciālvienādojumam ir tikai 30 soļi jeb 0.075s.

Tātad dziļo mākslīgo neironu tīklu modelis ir spējīgs gan precīzāk prognozēt nākamajā solī esošo dubultā svārsta daļu leņķisko ātrumu vērtības, gan ilgāk noturēt prognozes vērtības 8 rad/s intervālā no reālajām vērtībām. Turklāt statistiskā pārbaude apliecina šo izteikumu patiesumu - dziļo mākslīgo neironu tīklu modeļa MAE kļūdas vērtība ar ticamību 0.99 ir mazāka par labākās diferenciālvienādojuma metodes MAE kļūdu.

Tāpat arī iespējams veikt secinājumus starp dažādām LSTM modeļu arhitektūrām. Labākos rezultātus uzrādījušais dziļo mākslīgo neironu tīklu modelis izmanto visas papildus apskatītās modeļa daļas - rekurento daļu kļūdas aprēķinā ar pakāpenisku grūtību, kas apliecina apskatītajās publikācijās izvirzītos izteikumus par šo metožu lietderību. Turpretim modeļi ar vienādiem parametriem, bet atšķirīgu signāla retināšanu uzrāda sliktākus rezultātus, tātad konkrētajā eksperimentā šī metode nav noderīga.

Tomēr liels trūkums dziļo mākslīgo neironu tīklu metodēm ir to sarežģītība, ilgais laiks un resursi, kas jāiegulda to sagatavošanā un apmācībā. Labākā dziļo mākslīgo neironu tīklu modeļa apmācība RTU HPC centrā izmantojot K40 videokarti aizņem aptuveni 33 stundas. Tāpat jāņem vērā, ka labāko modeļa iestatījumu atrašanai jāapmāca daudzi modeļi, kas kopējās skaitļošanas resursu prasības paaugstina vēl vairāk.

Apzinātās īpašības ļauj iezīmēt arī praktiskus pielietojumus darbā apskatītajai dziļo mākslīgo neironu tīklu metodei - dažādu sistēmu diagnostikai, it īpaši modelējot sarežģītas situācijas, kurās iespējams ievākt lielu apjomu datu.

LITERATŪRAS SARAKSTS

1. Lagaris, I., Likas, A. & Fotiadis, D. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks* 9. sējums, 987.—1000. lpp. (1998. g. sept.).
2. Gannon, D. Deep Learning with Real Data and the Chaotic Double Pendulum (2021. g. febr.).
3. Raissi, M., Perdikaris, P. & Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* 378. sējums, 686.—707. lpp. ISSN: 0021-9991. https://www.sciencedirect.com/science/article/pii/S0021999118307125 (2019).
4. Alexis Asseman Tomasz Kornuta, A. S. O. *Learning beyond simulated physics* (2018).
5. M. Iltiņa, I. I. *Skaitliskās metodes* (RTU, Rīga, 2002).
6. Alexandrov, O. *Euler method* https://commons.wikimedia.org/wiki/File:Euler_method.svg.
7. Traum, H. *Runge-Kutta slopes* https://commons.wikimedia.org/wiki/File:Runge-Kutta_slopes.svg.
8. Arnold, F. W. Numerical Methods for Differential Equations, 12.—15. lpp. (2018).
9. Commons, W. *Neuron description* https://commons.wikimedia.org/wiki/File:Neuron.svg.
10. J.Zuters. *Neironu tīkli. Neironu tīklu uzbūve un darbība* (Rīga, 2010).
11. Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems* 25. sējums (2012. g. janv.).
12. Aggarwal, C. C. *Neural Networks and Deep Learning: A Textbook* 1st. ISBN: 3319944622 (Springer Publishing Company, Incorporated, 2018).
13. Nielsen, M. A. *Neural Networks and Deep Learning* http://neuralnetworksanddeeplearning.com/ (Determination Press, 2018).
14. Nabil, M. *Linear Regression With One Variable | Gradient descent* https://medium.com/@madali.nabil97/linear-regression-with-one-variable-gradient-descent-1dea65fef0a8.

15. Bento, C. *Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis* https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141.

16. Valtonen, V. *Elements of AI | Ievads par neironu tīkliem* https://course.elementsofai.com/lv/5/1.

17. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* http://www.deeplearningbook.org (MIT Press, 2016).

18. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Computation* 9. sējums, 1735.—1780. lpp. (1997).

19. Karpathy, A. *The Unreasonable Effectiveness of Recurrent Neural Networks* http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

20. Olah, C. *Understanding LSTM Networks* https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

21. Ioffe, S. & Szegedy, C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* 2015. https://arxiv.org/abs/1502.03167.

22. Ba, J. L., Kiros, J. R. & Hinton, G. E. *Layer Normalization* 2016. https://arxiv.org/abs/1607.06450.

23. Wu, Y. & He, K. *Group Normalization* 2018. https://arxiv.org/abs/1803.08494.

24. Goyal, A. *u. c. Professor Forcing: A New Algorithm for Training Recurrent Networks NIPS* (2016).

25. Graves, A., Bellemare, M. G., Menick, J., Munos, R. & Kavukcuoglu, K. Automated Curriculum Learning for Neural Networks. *CoRR* abs/1704.03003 sējums. arXiv: 1704.03003. http://arxiv.org/abs/1704.03003 (2017).

26. Bengio, S., Vinyals, O., Jaitly, N. & Shazeer, N. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *CoRR* abs/1506.03099 sējums. arXiv: 1506.03099. http://arxiv.org/abs/1506.03099 (2015).

27. Klinkachorn, S. & Parmar, J. *Evaluating Current Machine Learning Techniques On Predicting Chaotic Systems CS* (2019).

28. Rudy, S., Kutz, J. & Brunton, S. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics* 396. sējums (2019. g. jūl.).

29. Regazzoni, F., Dede, L. & Quarteroni, A. Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics* 397. sējums (2019. g. jūl.).

30. Chen, T. Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D. Neural Ordinary Differential Equations. *CoRR* abs/1806.07366 sējums. arXiv: 1806.07366. http://arxiv.org/abs/1806.07366 (2018).

31. Yao, W., Storey-Fisher, K., Hogg, D. W. & Villar, S. A simple equivariant machine learning method for dynamics based on scalars. *CoRR* abs/2110.03761 sējums. arXiv: 2110.03761. https://arxiv.org/abs/2110.03761 (2021).

32. Dufera, T. Deep neural network for system of ordinary differential equations: Vectorized algorithm and simulation. 5. sējums (2021. g. jūn.).

33. Savitzky, A. & Golay, M. J. E. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry* 36. sējums, 1627.—1639. lpp. https://pubs.acs.org/doi/abs/10.1021/ac60214a047 (1964).

34. Hafner, D. *Tips for Training Recurrent Neural Networks* Blog post. 2017. https://danijar.com/tips-for-training-recurrent-neural-networks/.

35. Misra, D. Mish: A Self Regularized Non-Monotonic Neural Activation Function. *CoRR* abs/1908.08681 sējums. arXiv: 1908.08681. http://arxiv.org/abs/1908.08681 (2019).

36. Ba, J., Kiros, J. & Hinton, G. E. Layer Normalization. *ArXiv* abs/1607.06450 sējums (2016).

37. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 2014. http://arxiv.org/abs/1412.6980.

38. Ruder, S. An overview of gradient descent optimization algorithms. *CoRR* abs/1609.04747 sējums. arXiv: 1609.04747. http://arxiv.org/abs/1609.04747 (2016).

39. Wang, X., Chen, Y. & Zhu, W. A Comprehensive Survey on Curriculum Learning. *CoRR* abs/2010.13166 sējums. arXiv: 2010.13166. https://arxiv.org/abs/2010.13166 (2020).

40. Virtanen, P. *u. c.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17. sējums, 261.—272. lpp. (2020).