Kauno technologijos universitetas
Matematikos ir gamtos mokslų fakultetas
Informatikos fakultetas

Rekurentinio neuroninio tinklo apmokymas panaudojant grafinį vaizdo procesorių ir tinklo pritaikymas labiausiai tikėtinų žodžių pasiūlymui duotai sakinio pradžiai

Baigiamasis bakalauro projektas

Deividas Riabčinskis
Projekto autorius

asist. Mindaugas Bražėnas
Vadovas

doc. Agnė Paulauskaitė-Tarasevičienė
Vadovė

2019, Kaunas

Kauno technologijos universitetas
Matematikos ir gamtos mokslų fakultetas
Informatikos fakultetas

Rekurentinio neuroninio tinklo apmokymas panaudojant grafinį vaizdo procesorių ir tinklo pritaikymas labiausiai tikėtinų žodžių pasiūlymui duotai sakinio pradžiai

Baigiamasis bakalauro projektas

Deividas Riabčinskis
Projekto autorius

Taikomoji matematika (612G10002)
pagrindinės krypties studijos

Informatika (612I10004)
gretutinės krypties studijos

asist.
Mindaugas Bražėnas
Vadovas

doc.
Agnė Paulauskaitė-Tarasevičienė
Vadovė

lekt. dr.
Mantas Landauskas
Recenzentas

doc.
Martynas Patašius
Recenzentas

2019, Kaunas

Kauno technologijos universitetas
Matematikos ir gamtos mokslų fakultetas
Deividas Riabčinskis

Rekurentinio neuroninio tinklo apmokymas panaudojant grafinį vaizdo procesorių ir tinklo pritaikymas labiausiai tikėtinų žodžių pasiūlymui duotai sakinio pradžiai

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Liepos Liepaitės, baigiamasis projektas tema „Baigiamojo projekto pavadinimas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____	_____
(vardą ir pavardę įrašyti ranka)	(parašas)

Riabčinskis, Deividas. Rekurentinio neuroninio tinklo apmokymas panaudojant grafinį vaizdo procesorių ir tinklo pritaikymas labiausiai tikėtinų žodžių pasiūlymui duotai sakinio pradžiai. Bakalauro baigiamasis projektas / vadovas asist. Mindaugas Bražėnas; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): pagrindinų studijų - Taikomoji matematika (Fiziniai mokslai) ir gretutinių studijų - Informatika (Fiziniai mokslai).

Reikšminiai žodžiai: neuronas, rekurentinis, gradientas, tinklas, apmokymas.

Kaunas, 2019. 30p.

Santrauka

Baigiamojo projekto darbo tikslas susidaryti ir išsivesti rekurentinio neuroninio tinklo veikimo formules ir tas formules realizuoti programiškai. Įvade aptariamas projekto aktualumas, suformuluojama problema ir iškeliami darbo uždaviniai.

Literatūros analizės skyriuje nagrinėjami įvairūs neuroniai tinklai, šių tinklų veikimo principai, sandara ir jų realizavimo metodai. Aptarsiu kokios paprastesnių tinklų savybės yra naudojamos, kuriant vis sudėtingesnį neuroninį tinklą iki mano tiriamo LSTM rekurentinio neuroninio tinklo.

LSTM rekurentinio neuroninio tinklo skyriuje bus tiksliai paaiškintas mano pasirinkto neuroninio tinklo veikimos principas, aprašyti naudojami kintamieji, sudarytos ir išvestos formulės reikalingos tinklui apmokinti ir prognozuoti reikšmes.

Tiriamojoje dalyje bus analizuojamas tinklo veikimas šį tinklą apmokinant duotu tekstu. Tinklo veikimas bus tiriamas stebint paklaidos funkcijos reikšmės kitimą didėjant apmokymų skaičiui. Taip pat bus tiriama programos metodų greitaveika, kaip greitai tinklas atlieka skaičiavimus esant skirtingoms topologijoms.

Riabčinskis, Deividas. Recurrent neural network training using GPU and its application for suggesting the most probable words given the beginning of sentence. Bachelor's Final Degree Project / supervisor assoc. asistant Mindaugas Bražėnas; The Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): main - Applied Mathematics (Physics studies), secondary - Informatics (Physics studies).

Keywords: neuron, recurrent, gradient, network, teaching.

Kaunas, 2019. 30p.

Summary

Bachelor's Final Degree Project's main goal is to create and derive recurrent neural network formulas and write a program to realise these formulas. In the introduction I will discuss this project's relevance, formulate the problem of this project and raise the tasks.

In the literature analysis, I will discuss diffrent types of neural networks, how these networks work and the way they can be realised. I will talk about what simplier neural network properties are used to create more difficult neural networks.

In the LSTM recurrent neural network section I will accurately explain my selected neural network's working principles, it's structure, describe used variables, create and derive formulas which are required to teach the network and to output the output values based on the inputs. I will also explain network's application to the selected problem and describe this network's implementation using programming language C++.

In the investigatory section I will analyze how this network works training it with the given text. How this network works will be investigated by observing error function values change over number of training done. I will also investigate program's calculations speed using diffrent topologies.

TURINYS

PAVEIKSLAI

Įvadas

Šiais laikais labai daug yra kalbama apie dirbtinį intelektą, robotus, kurie jau dabar yra naudojami padedant žmonėm atlikti kasdienius darbus ir netgi robotai keičia žmones gamyklose atliekant specifinius darbus.

Šiuolaikinėje visuomėneje, turbūt nei vienas negali įsivaizduoti gyvenimo be telefono ar mašinos. Naudojant dirbtinį intelektą yra kuriamos telefoninės aplikacijos, kurios leidžia siųsti sms žinutes jų nerašant, o pasakant žinutės tekstą balsu, taip pat muzikinės aplikacijos, kurios padeda surasti dainos pavadinimą išgirdus melodijai ir netgi dirbtinis intelektas yra naudojamas kaip vertėjas, gali du žmonės šnekėti skirtingomis kalbomis, bet naudojant dirbtinį intelektą, žodžiai iškart yra išverčiami į gimtąją kalbą. Kompanijos TESLA yra kuriami automobiliai, kurie turi įrangą, kuri leidžia automobiliams patiems važiuoti.

Dirbtinis intelektas yra kuriamas naudojant visokiausius neuroninius tinklus. Šie tinklai veikia panašiai kaip žmogaus smegenys, kaip žmogaus smegenų ląstelės. Šių tinklų yra įvairiausių rūšių, nuo paprastų neuroninių tinklų iki rekurentinių neuroninių tinklų.

Projekto tikslas - sukurti neuroninį tinklą, kuris būtų galimas pritaikyti praktikoje. Mano tikslas šį tinklą pritaikyti taip, kad jis išmoktų pasiūlyti, žmogui rašančiam tekstą, pradėto rašyti žodžio galą ir sekantį žodį.

Darbo uždaviniai:

1. Surasti informacijos apie rekurentinius neuroninius tinklus ir suprasti jų veikimą.
2. Išsivesti rekurentinio neuroninio tinklo, LSTM (angl. *Long Short Term Memory*) apmokymo formules.
3. Optimizuoti formules.
4. Formules realizuoti programiškai.
5. Testuoti programos fragmentus.
6. Pritaikyti tinklą prognozuoti rašomo žodžio galą ir sekantį žodį.
7. Atlikti suprogramuoto rekurentinio neuroninio tinklo veikimo analizę.

1. Literatūros apžvalga

1.1. Neuroniniai tinklai

Neuroniai tinklai arba kitaip dirbtinio intelekto neuroniai tinklai yra mašininio mokymo metodai, kurie leidžia kompiuteriui mokytis iš stebimų duomenų. Neuroninių tinklų veikimo principas yra įkvėptas pagal tai kaip biologinė nervų sistema apdoroja informaciją.[1]

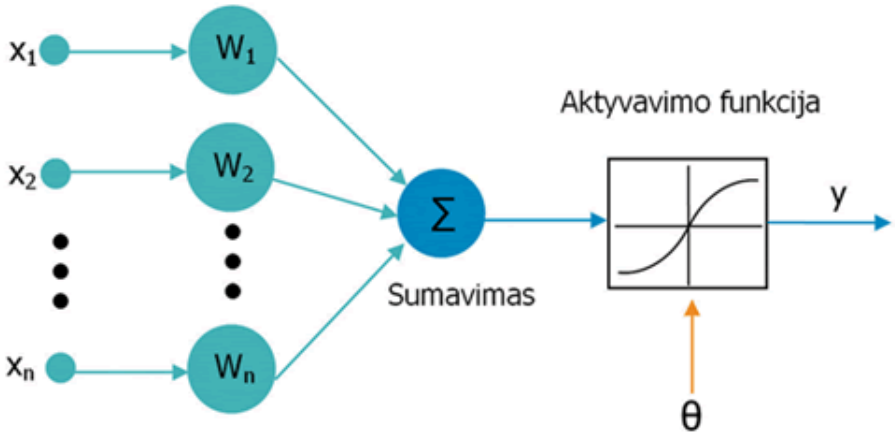
Neuroniai tinklai dažniausiai yra naudojami išvesti prasmę iš sudėtingų netiesinių duomenų, nustatyti ir ištraukti modelius, kurie yra sunkiai pastebimi ar net išvis neįmanoma nustatyti modelio, kuris aprašytų esamus duomenis.[1]

1.2. Neuroninių tinklų neuronai

Esminis neuroninio tinklo elementas yra dirbtinis neuronas. Šį neuroną sudaro pagrindiniai trys komponentai: įvesties reikšmės, svoriai, kurie jungia šias reikšmes su neuronu ir aktyvacijos funkcija. 1 paveiksle pavaizduotas yra pats paprasčiausias neurono modelis, vadinamas Perceptronu (angl. *Perceptron*).[2]

1 pav. Perceptronas.

Šaltinis: https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464



2 pav. Dirbtinio neurono modelis (Perceptronas).

Šaltinis: http://www.elektronika.lt/teorija/kompiuterija/4342/neuroniniai-tinklai/

Paveiksle pavaizduoto Perceptrono reikšmė apskaičiuojama pagal formulę

$$y = f\left(\sum_{k=1}^{n} x_k W_k\right),$$

čia f - aktyvavimo funkcija.

Paveiksle 2 pavaizduoto neurono veikimo principas:

1. Neuronas gauna įvesties reikšmes $x_1, x_2, ..., x_n$. Kiekviena iš reikšmių turi savo perdavimo koeficientą į neuroną (svorį) $W_1, W_2, ..., W_n$.
2. Skaičiuojama įvesties reikšmių ir atitinkamų svorių sandaugų suma (žymima z).

$$z = \sum_{k=1}^{n} x_k W_k$$

3. Neurono išvesties reikšmė (žymima a) yra apskaičiuojama įvesties reikšmių ir atitinkamų svorių sandaugų sumai pritaikius aktyvacijos funkciją (žymima f).

$$a = f(z) = f(\sum_{k=1}^{n} x_k W_k)$$

1.3. Sužadinimo signalai

Dažniausiai įvesties reikšmių yra vienetu daugiau, nei yra paduodama įvesties reikšmių. Ši įvestis yra vadinama sužadinimo signalu (angl. *bias neuron*). Šio signalo reikšmė yra pastovi ir visada lygi vienetui($x_{n+1} = 1$). Taip pat yra pridedamas papildomas svoris W_{n+1} jungiantis šį signalą su neuronu.[3]

Šis sužadinimo signalas padeda lengvai koreguoti neurono išvesties reikšmę, kadangi įvesties reikšmė yra vienetas, tuomet sandauga, kuri yra prisumuojama priklauso tik nuo svorio jungiančio sužadinimo signalą su neuronu. Ji yra lygi W_{n+1}. Naudojant sužadinimo signalus neuroniai tinklai yra daug sparčiau apmokinami, kadangi šie signalai gali laisvai koreguoti perduodamą reikšmę į neuronus.

1.4. Aktyvacijos funkcijų savybės ir naudojimas

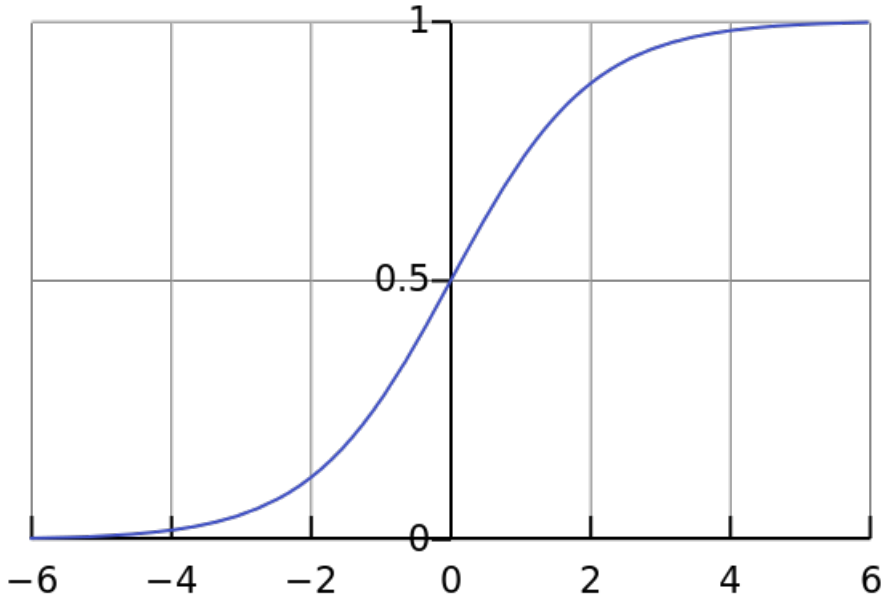
Neurono išvesties reikšmei apskaičiuoti naudojamų aktyvacijos funkcijų yra labai daug. Jų paskirtis yra nuspręsti ar atitinkamas neuronas, kuriam yra taikoma aktyvacijos funkcija yra reikšmingas tolimesniem skaičiavimams.

Tarkime turime neuroną, kurio z yra lygus $\sum_{k=1}^{n} x_k W_k + bias$. Šios reikšmės reikšmių intervalas yra nuo $-\infty$ iki $+\infty$. Kadangi neuronai nežino reikšmių ribų, tam tikslui yra naudojamos aktyvacijos funkcijos nuspręsti ar šis neuronas turi būti sužadintas.

Aktyvacijos funkcijos negali būti tiesinės. Straipsnyje rašoma, kad tarkim turint tiesinę funkciją f(x)=cx, mes ją galime naudoti skaičiuojant kai kurių neuronų reikšmes, tačiau bendruoju atveju, ka-

dangi neuronių tinklų tikslas yra tobulėti (t.y. mokytis). Neuroninių tinklų mokymas vyksta, naudojant gradientinio nusileidimo metodą. Šiam metodui yra reikalingos tinkluose naudojamos aktyvacijos funkcijos išvestinė. Kadangi šios tiesinės funkcijos išvestinė yra lygi konstantai, tai baudos funkcijos gradientas irgi bus lygus konstantai. Kadangi svorių atnaujinimas vyks priešinga gradiento kryptimi pagal konstantą, tai tinklas nebus apmokomas, nes funkcijos gradientas nepriklausys nuo įvesties reikšmių. [4]

Sigmoidinė netiesinė aktyvacijos funkcija. Jos funkcija $f(x) = \frac{1}{1+e^{-x}}$ ir grafikas paveiksle 3.



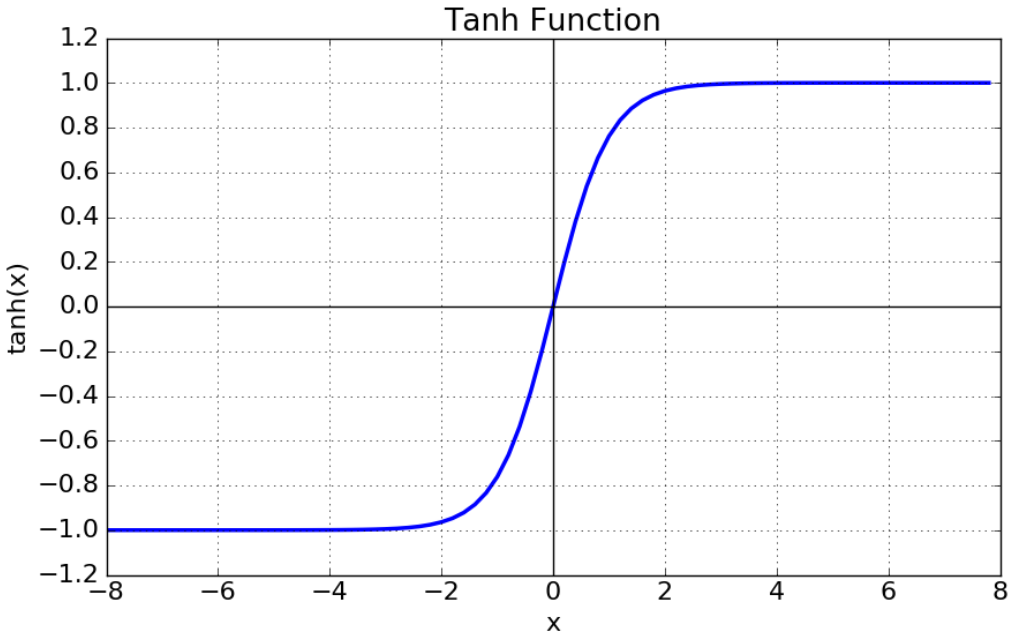
3 pav. Sigmoidinės funkcijos grafikas.

Šaltinis: https://en.wikipedia.org/wiki/File:Logistic-curve.svg

Įvardinsime šios aktyvacijos funkcijos privalumus ir trūkumus. [4]

1. Privalumai:
   (a) Netiesinė funkcija.
   (b) Šios funkcijos išvestinė taip pat netiesinė funkcija.
   (c) Šios funkcijos reikšmių intervalas yra nuo 0 iki 1, tai reiškia, kad funkcijos reikšmės turi rėžius, palyginus su tiesinėmis funkcijomis, kai rėžiai buvo $-\infty$ iki $+\infty$.
   (d) Kai x kinta nuo -2 iki 2, tai funkcijos reikšmės yra labai stačios(mažas x pakitimas turi daug įtakos funkcijos reikšmei), o kitais atvejais funkcija gražina arba labai žemą arba labai aukštą reikšmę, kas leidžia išskirti ryškias savybes.
2. Trūkumai:
   (a) Kai funkcijos gražinamos reikšmės yra arti 0 arba 1, tai funkcijos reikšmės labai mažai pasikeičia, kai kinta x. tai reiškia, kad gradiento reikšmė toje srityje bus maža. Tai gali sukelti gradiento išnykimą (angl. *vanishing gradient*), to pasekoje tinklas mokosi labai lėtai arba išvis nesimokina, jei neįvyksta jokiu drąstinių pokyčių.

Hiperbolinio tangento netiesinė aktyvacijos funkcija. Jos funkcija $f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$ ir grafikas paveiksle 4.



4 pav. Hiperbolinio tangento grafikas.

Šaltinis: https://mlnotebook.github.io/post/transfer-functions/

Įvardinsime šios aktyvavimo funkcijos privalumus ir trūkumus. [4]

1. Privalumai:
   (a) Netiesinė funkcija.
   (b) Šios funkcijos išvestinė taip pat netiesinė funkcija.
   (c) Šios funkcijos reikšmių intervalas yra nuo -1 iki 1, tai reiškia, kad funkcijos reikšmės turi rėžius.
   (d) Hiperbolinio tangento gradientai yra stipresni, nei sigmoidinės funkcijos(t.y. išvėstinės yra statesnės).
2. Trūkumai:
   (a) Kaip ir sigmoidinėje funkcijoje hiperbolinis tangentas turi tą pačią nykstančio gradiento problemą.

1.5. Tiesioginio sklidimo neuroniniai tinklai

Naudojant Perceptronų sudarymo logiką, buvo pradėti kurti pradiniai tiesioginio sklidimo neuroniai tinklai. Šie tinklai yra sudaryti iš daug perceptronų, kurie yra sugrupuoti į tris sluoksnius (5).

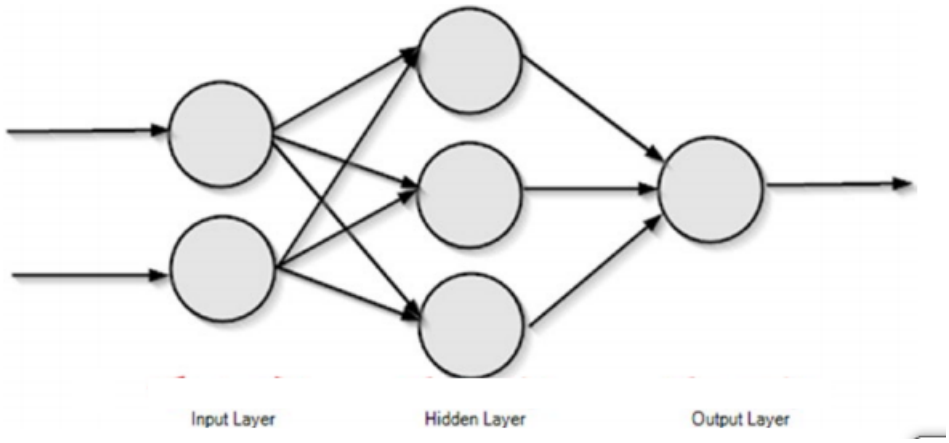
1. Įvesties sluoksnis (angl. *Input layer*).
2. Paslėptasis sluoksnis (angl. *Hidden layer*).

3. Išvesties sluoksnis (angl. *Output layer*).

Įvesties sluoksnis. Įvesties sluoksnis sudarytas iš neuronų, kurie priima įvesties reikšmes. Šio sluoksnio neuronų reikšmės yra tokios pat kaip ir įvesties reikšmės. [5]

Išvesties sluoksnis. Išvesties sluoksnis yra paskutinis šio neuroninio tinklo sluoksnis, kuris vartotojui gražina tinklo apskaičiuotas reikšmes. [5]

Paslėptasis sluoksnis. Paslėptasis sluoksnis yra išdėstytas taip, kad į jį įeinantys neuronai yra įvesties sluoksnio neuronai, o šio paslėptojo sluoksnio neuronai įeina į išvesties sluoksnio neuronus. Tai yra tarpinis sluoksnis tarp įvesties ir išvesties sluoksnių. [5]



5 pav. Tiesioginio sklidimo neuroninis tinklas.

Šaltinis: https://dzone.com/articles/feed-forward-neural-network-with-mxnetr

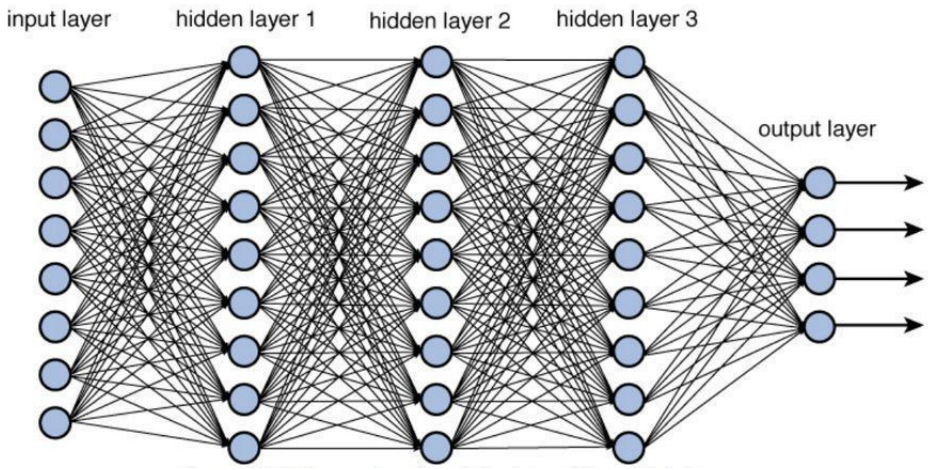
Paveiksle 5 galima pastebėti, kad

1. Du gretimi esantys sluoksniai yra pilnai sujungti svoriais.
2. Reikšmių apskaičiavimas vyksta pradedant įvesties sluoksniu, toliau reikšmes perduodant paslėptajam sluoksniui ir pabaigoje iš paslėptojo sluoksnio reikšmes perduodant išvesties sluoksniui.
3. Tarp įvesties ir išvesties sluoksnių yra vienas paslėptasis sluoksnis.

1.6. Daugiasluoksniai tiesioginio sklidimo neuroniniai tinklai

Turintys daugiau nei vieną paslėptąjį sluoksnį, tiesioginio sklidimo neuroniniai tinklai yra vadinami daugiasluoksniais. Šio tinklo veikimo principas yra toks pat, kaip ir paprastojo tiesioginio neuroninio tinklo su vienu paslėptuoju sluoksniu. Šių tinklų privalumas yra tas, kad kiekvienas paslėptasis sluoksnis nustato tam tikras įvesties duomenų charakteristikas [6]. Pavyzdžiui [7] darbe rašoma, kad atliekant šį tinklą apmokant atpažinti ranka rašytinius skaičius, tinklo paslėptieji sluoksniai nusako tam tikras skačiaus charakteristikas, pavyzdžiui vienas sluoksnis gali nusakyti ar skaičiuje yra apskritimas, tai jei sluoksnis atranda apskritimą tame skaičiuoje, tai tinklo gražinamos reikšmės gali būti 0, 6, 8 arba 9, kadangi šiuose skaičiuose yra apskritimas.

Norint žinoti kiek sluoksnių yra šiuose neuroniniuose tinkluose, skaičiavimams atlikti yra įvedamas žymėjimas L, kuris reiškia kiek sluoksnių šitame neuroniniame tinkle yra, o žymėjimas l reiškia kuriame sluoksnyje dabar atliekame skaičiavimus.



6 pav. Daugiasluoksnis tiesioginio sklidimo neuroninis tinklas.

Šaltinis: https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

Šio tinklo bet kurio neurono skaičiavimo formulė.

$$a_k^l = f(\sum_{n=1}^{s_{l-1}} a_n^{l-1} W_{nk}^{l-1} + bias),$$

čia s_l - nurodo kiek l-ajame sluoksnyje yra neuronų, a_k^l - nurodo k-ąjį neuroną l-ajame sluoksnyje, o W_{nk}^{l-1} - nurodo svorį jungiantį (l-1)-ojo sluoksnio n-ąjį neuroną su l-ojo sluoksnio k-uoju neuronu.

1.7. Rekurentiniai neuroniniai tinklai

Rekurentiniai neuroniai tinklai implementuoja naujo tipo ląsteles, vadinamas rekurentinėmis ląstelėmis. Pirmasis šio tinklo prototipas buvo sukurtas, taip kad kiekviena paslėpta tinklo ląstelė kaip įvestis pasiimdavo tos pačios ląstelės išvesties reikšmes kelių žingsnių. [8]

Yra daug šio tinklo variacijų, pavyzdžiui kaip įvestis perduoti tinklo būsenas, kintamųjų užtempimas ir panašiai, tačiau įdėja išlieka ta pati. Šis tinklas dažniausiai yra naudojamas, kai yra svarbus kontekstas (t.y. kai sprendimai priklauso nuo to kas vyko praeityje), pavyzdžiui sudarant tekstus, nauji teksto sakiniai yra prognozuojami, pagal tai kokie sakiniai jau yra prieš tai parašyti. [9]

1.8. LSTM tipo rekurentiniai neuroniniai tinklai

LSTM - Ilgalaikė/trumpalaikė atmintis. Šio tipo rekurentiniai tinklai implementuoja atminties ląstelę, kuri gali apdoroti duomenis, kai paduodami duomenys turi tam tikrus laiko tarpus, kada jie yra paduodami. Šie tinklai gali prognozuoti tekstą pagal pavyzdžiui 10 prieš tai einančių žodžių. Taip pat šie tinklai gali prognozuoti vaizdo kadrą, pagal tai kas vyko keliais kadrais ankščiau. [10]

Atminties ląstelės yra sudarytos iš kelių elementų vadinamų vartais(angl. *gates*), kurie kontroliuoja kaip informacija yra prisimenama ir pamirštama.

Mūsų darbe naudojamo tinklo vartų ląstelių yra keturios:

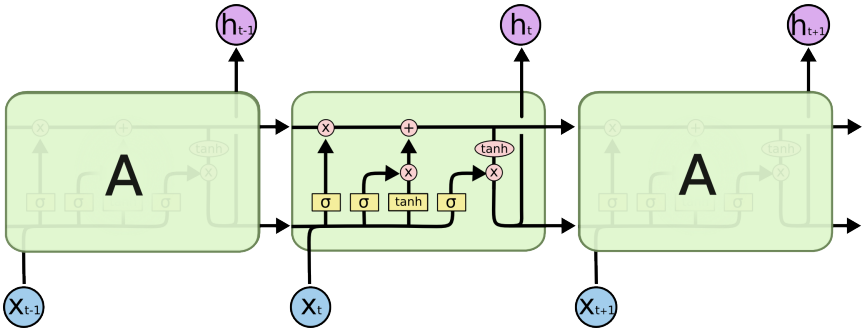
1. Pamiršimo vartai (angl. *forget gate*).

2. Įvesties vartai (angl. *input gate*).
3. Įvesties moduliacijos vartai (angl. *input modulation gate*).
4. Išvesties vartai (angl. *output gate*).

Kiekviena iš keturių tinklo ląstelių yra sudaryta iš daugiasluoksnių tiesioginių neuroninių tinklų. Kiekviena iš ląstelių atlieka savo vaidmenį šiame tinkle.

1. Pamiršimo ląstelė nurodo, kuri perduodama informacija turi būti pamirštama, nereikšminga.
2. Įvesties ląstelė kontroliuoja kiek naujos informacijos iš įvesties reikšmių reikia pridėti prie jau esamos informacijos.
3. Įvesties moduliacijos ląstelė papildomai pakoreguoja kiek naujas informacijos reikia pridėti prie jau esamos informacijos.
4. Išvesties ląstelė nurodo, kuri informacija yra svarbi ir kiek jos turi būti perduodama tolimesniems skaičiavimam.

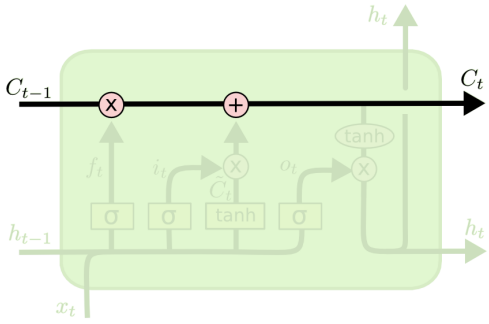
Šių visų tinklų perduodama informacija sąveikauja tarpusavyje (7 pav.).



7 pav. LSTM tinklo sandara.

Šaltinis: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pagrindinė šio tinklo veikimo esmė yra atmintis, kuri yra perduodama kaip būsena prognozuojant naujas tinklo išvesties reikšmes.



8 pav. LSTM tinklo atmintis.

Šaltinis: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

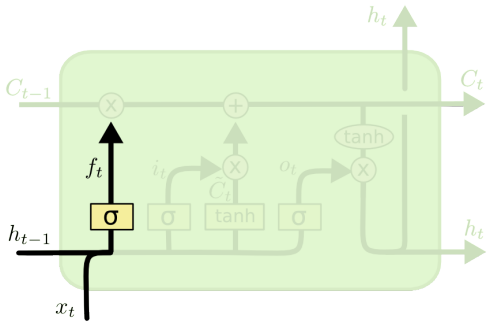
LSTM tinklai turi galimybę ištrinti arba įrašyti informaciją į atmintį. Kuri informacija yra perduodama

arba pamirštama yra kontroliuojama ląstelių, kurios yra neuroniniai tinklai su individualiomis charakteristikomis.[10]

Rekurentinis neuroninis tinklas turi tris neuroninius tinklus, kurie naudoja sigmoidinę aktyvacijos funkciją ir vieną tinklą, kuris naudoja hiperbolinio tangento aktyvacijos funkciją.

LSTM veikimo principas [10]

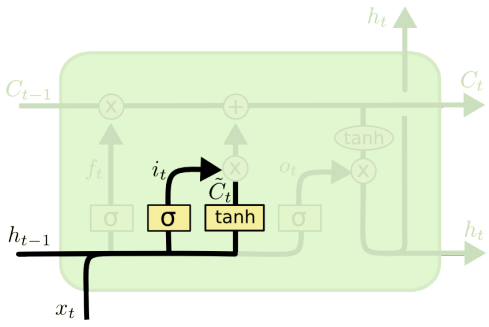
1. Pirmiausia nustatome, kurią informaciją turime pamiršti. Šį sprendimą daro pamiršimo ląstelė. Ji pagal tinklo būseną ir įvesties reikšmes nustato, kurią informaciją reikia visiškai pamiršti, o kurią reikia perduoti. (9 pav.)



9 pav. LSTM tinklo pamiršimo ląstelė.

Šaltinis: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

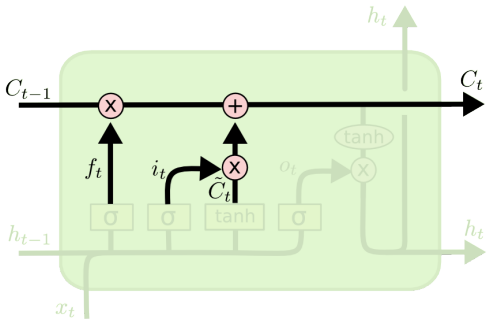
2. Sekantis žingsnis yra nustatyti kurią informaciją įrašysime į atmintį. Šį sprendimą daro įvesties ir įvesties moduliacijos ląstelės. Įvesties ląstele nustatome, kurią informaciją mes pridėsime, o įvesties moduliacijos ląstele nustatome kokia ta informacija bus. (10 pav.)



10 pav. LSTM tinklo įvesties ląstelės.

Šaltinis: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

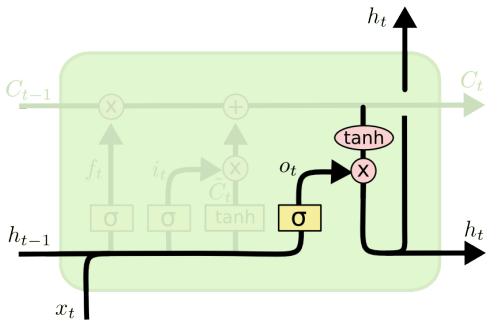
3. Toliau atnaujinsime tinklo atmintį. Iš pradžių padauginame atmintį iš pamiršimo ląstelės, kad pamirštume informaciją, kurią nustatėme, kad reikia pamiršti. o toliau pridėsime informaciją, kurią nustatė įvesties ir įvesties moduliacijos ląstelės, kad reikia pridėti.(11 pav.)



11 pav. LSTM tinklo naujos atminties skaičiavimas.

Šaltinis: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

4. Galiausiai nustatome kokią tinklo reikšmę išvesime. Ši reikšmė priklausys nuo naujos atminties ir išvesties ląstelės. Išvesties ląstelė nustato kurias reikšmes reikia perduoti, o atminčiai yra pritaikoma hiperbolinio tangento funkcija. Sudauginus atmintį ir išvesties ląstelę, gauname kurią informaciją išvesime.(12 pav.)



12 pav. LSTM tinklo naujos atminties skaičiavimas.

Šaltinis: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Tačiau mano kuriamame tinkle išvesties reikšmė dar nebus galutinė. Šiuo atveju naudosiu softmax funkciją, kuri išvesties reikšmes normalizuos į tikimybinį pasiskirstymą. Ši funkcija yra naudojama, kadangi išvesties reikšmių intervalas yra nuo -1 iki 1 ir šių reikšmių suma gali būti nelygi vienetui, tačiau pritaikius šią funkciją išvesties reikšmių intervalas yra nuo 0 iki 1 ir išvesties reikšmių suma yra lygi vienetui, tam kad būtų galima patogiau suprasti tinklo išvestį.

Rekurentinio tinklo apmokymas vyksta klaidos sklidimo atgal metodu(angl. *back propogation*).

Tinklo apmokymo principas

Tinklo apmokymas grindžiamas minimizuojant tinklo baudos funkcijos išvestinę pagal visus tinkle esančius svorius.

$$\frac{\partial E}{\partial W_{ij}}$$

Naudojant šią išvestinę tinklo apmokymas vykdomas atnaujinant visus tinklo svorius.

$$W_{ij} = W_{ij} + \frac{\partial E}{\partial W_{ij}}$$

Tačiau toks tinklo apmokymo būdas nėra veiksmingas, nes tinklas apsimokys atpažinti naujas reikšmes labai greitai, tačiau problema yra ta, kad labai lengvai pamirš ką jau išmoko praeityje, todėl yra naudojami α ir η koeficientai. Gaunama nauja formulė.

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} + \alpha \Delta W_{ij}$$

$$W_{ij} = W_{ij} + \Delta W_{ij}$$

α ir η kintamieji nurodo tinklo apmokymo inerciją ir greitį. [7] darbe lyginamos tinklo apmokymo paklaidos keičiant šių parametrų reikšmes. Parinkus optimalius parametrus tinklo svoriai daug greičiau konverguoja į norimus svorius, kas leidžia tinklui greičiau apsimokinti. Parinkus labai mažas apmokymo greičio reikšmes tinklas gali labai užtrukti, kol jis apsimokins, o parinkus labai dideles reikšmes tinklas labai greitai išmoks atpažinti naujas reikšmes, tačiau pamiršdamas senas.

Taip pat egzistuoja nykstančio gradiento problema, kuri būna tinkluose, kuriuose yra labai daug sluoksnių. Kadangi neuroninių tinklų apmokymas vyksta einant nuo paskutinio sluoksnio iki pirmojo ir taip einant per visus sluoksnius kiekvieno sluoksnio išvestinės yra dauginamos kartu, tam kad apskaičiuotų pradinio sluoksnio išvestines. Kai yra n neuroninio tinklo sluoksnių ir šie sluoksniai naudoja tokią aktyvacijos funkciją, kaip sigmoidinę, tai n sluoksnių išvestinės yra sudauginamos kartu, to pasekoje gradiento reikšmė mažėja eksponentiškai, o maža gradiento reikšmė reiškia, kad apmokant tinklą svoriai nebus atnaujinami, kadangi gradiento reikšmė, pagal kurią atnaujiname svorius bus nulinė.[11] [12]

1.9. Užklausų automatinis užbaigimas

Neuroniniai tinklai yra taikomi daugelyje sričių. Viena iš jų yra kalbos sudaryme (angl. *language modelling*). Daugelis šiuolaikinių paieškos sistemų naudoja užklausų automatinio užbaigimo sistemas. Šios sistemos siūlo rašomų žodžių pabaigą ir prognozuoja kelis sekančius žodžius. Šios sistemos veikimo principas yra, kai vartotojas įvedą į paiešką bet kokią raidę sistema pagal prieš tai buvusį tekstą ir pagal naują įvestą raidę pasiūlo naujus žodžių užbaigimo variantus ir prognozuoja naujus sekančius žodžius. Tokie pasiūlymai dažniausiai yra priimami iš praeities naudotų užklausų, todėl yra labai mažas šansas, kad sistema gebės prognozuoti tekstą, kurio dar nėra mačius. Tam tikslui yra naudojami neuroniai tinklas, kad apmokyti tinklą prognozuoti tokių užklausų užbaigimus.[13]

1.10. Programnės įrangos apžvalga

Neuroniniai tinklai yra programuojami daugeliu programavimo kalbų, tokių kaip Python, C++, C#, R, Java ir kt. Mano pasirinkta kalba yra C++, kadangi ši kalba yra pakankamai paprasta ir palaiko objektiškai orientuotą programavimą. Darbe rekurentinis neuroninis tinklas bus objektas, kuris turės keturis paprastus neuroninius tinklus, kurie taip pat bus atskiri objektai.

Neuroninio tinklo išvestoms formulėms optimizuoti buvo naudojamas vienas iš algoritmų analizės metodų. Naudojau dinaminio programavimo metodą. Šis metodas rekurentinio uždavinio sąlygą suskaido į kelias mažesnes sąlygas. Dinaminio programavimo implementacijos būdų yra keli, pagrindiniai jų yra metodas, kuris skaičiuoja iš pradžių mažesnių uždavinių reikšmes, jas išsaugo, poto tas reikšmes apjungia ir skaičiuoja sudėtingesnio uždavinio reikšmes. Kitas būdas skaičiuojant didesnio

uždavinio reikšmę, tai didesnis uždavinys suskaidomas į kelis mažesnius uždavinius ir tie uždaviniai yra sprendžiami atskirai. Šiame darbe naudosiu pirmąjį būdą, nes visus skaičiavimus galima išsaugoti atmintyje ir jie neužima daug vietos, o kadangi daug sudėtingesnių uždavinių sprendimai priklausys nuo dar daugiau paprastesnių uždavinių, tai šis būdas yra optimaliausias greitaveikos atžvilgiu. [14]

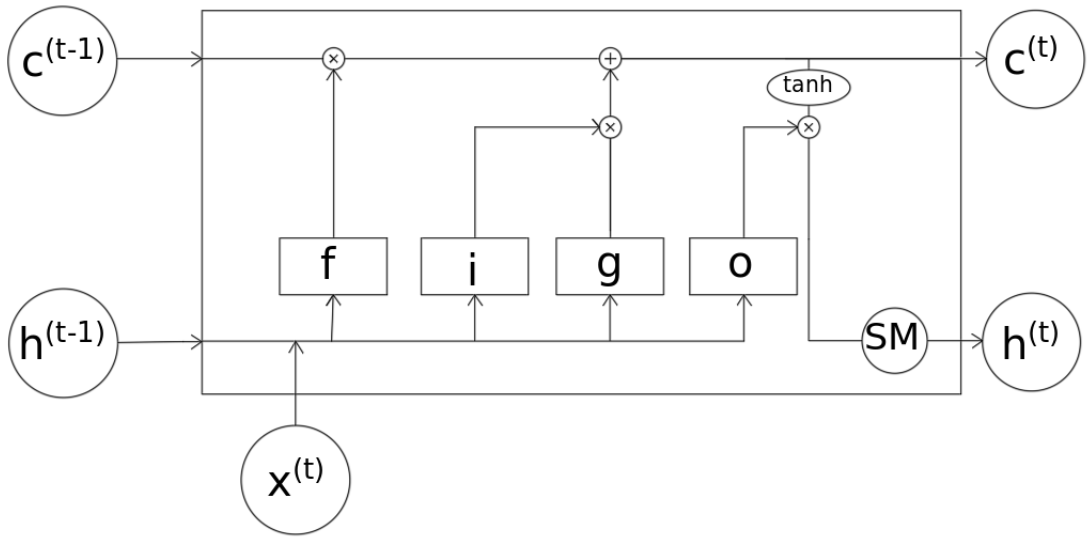
Tinklo apmokymas ant procesoriaus vyksta lėtai, kadangi tinklui apmokyti reikia atlikti labai daug skaičiavimų, todėl skaičiavimai bus perkeliami ant vaizdo procesoriaus (angl. *GPU*). Šiam tikslui yra naudojamos lygiagretaus programavimo bibliotekos. C++ kalboje dažniausiai naudojamos yra OpenMP ir CUDA. Darbe naudosiu CUDA biblioteką, kadangi ji yra pritaikyta skaičiavimus perkelti į grafinį vaizdo procesorių ir juos paskirstyti. Taip pat skaičiavimai yra atliekami daug greičiau grafiniame vaizdo procesoriuje, nes jame yra tūkstančiai gijų, kurios nepriklausomus skaičiavimus gali atlikti vienu metu. [15]

2. LSTM rekurentinis neuroninis tinklas

2.1. LSTM sandara

Kuriamo neuroninio tinklo sandara bus tokia pati į literatūroje apžvelgtą LSTM rekurentinio neuroninio tinklo, tačiau detaliau aprašysiu visą tinklo veikimą ir naudojamus kintamuosius.

Sudarytas LSTM rekurintinis neuroninis tinklas. (13 pav.)



13 pav. Rekurentinis neuroninis tinklas.

M - išvesties ir atminties vektorių ilgis.
I - įvesties reikšmių vektoriaus ilgis.
V - neuroninių tinklų kiekis.
L - nurodo atitinkamo neuroninio tinklo paskutinįjį sluoksnį.
u - nuorodo skaičiuojamąjį tinklą.
v - nuorodo bet kurį tinklą iš visų esamų (t.y. nėra skirtumo pagal, kurio tinklo reikšmės skaičiavimai yra atliekami).

Atmintis. Tinklas turi atmintį, kuri yra žymima $c_k^{(t)}$, čia c - vektorius, kuris saugo t laiko momentu gautas tinklo atminties reikšmes. Jo dydis yra M.

Tinklo būsena. Tinklas praeito žingsnio išvestį perduodą kaip naują būseną į dabartinį laiko momentą. Ji žymima $h_i^{(t)}$, čia h - vektorius, kuris saugo t laiko momentu gautas tinklo išvesties reikšmes. Jo dydis yra M.

Įvesties reikšmės. Esamuoju laiko momentu paduotos įvesties reikšmės žymimos $x_i^{(t)}$. Šio vektoriaus ilgis yra I.

Neurono tarpinė reikšmė. Į neuroną paduodamų sandaugų suma žymima $z_k^{u,l}$, čia z - nurodo u-ojo tinklo, l-ojo sluoksnio, k-ojo neurono sumą.

Neurono reikšmė. Neurono reikšmė pritaikius neurono tarpinei reikšmei aktyvacijos funkciją žymima $a_k^{(u,l)}$, čia a - nurodo u-ojo tinklo, l-ojo sluoksnio, k-ojo neurono reikšmę.

Neuroniai tinklai turi du svorių rinkinius:

1. **Neuronus jungiantys svoriai**. Svoriai jungiantys dviejų gretimų sluoksnių neuronus žymimi $w_{ij}^{(v,s)}$, čia W - nurodo v-ojo tinklo svorį iš s-ojo sluoksnio i-ojo neurono į (s+1)-ojo sluoksnio j-ąjį neuroną.
2. **Būsenų svoriai**. Svoriai jungiantys tinklo būseną su tinklų antrojo sluoksnio neuronais žymimi $w_{ij}^{(v)}$, čia W - nurodo v-ojo tinklo svorį iš $h_i^{(t)}$ būsenos į 2-ojo sluoksnio j-ąjį neuroną.

Neuronų kiekiai sluoksniuose. Vektorius saugantis neuroninių tinklų atitinkamų sluoksnių neuronų kiekis žymimas $K(u,l)$, čia K - nurodo u-ojo tinklo, l-ojo sluoksnio neuronų kiekį.

Tarpinės rekurentinio tinklo reikšmės žymimos $b_k^{(t)}$, čia b - nurodo tarpines k-asias išvesties reikšmes laiko momentu t.

Tinklo baudos funkcija. Tinklo paklaidą skaičiuojanti funkcija žymima $E^{(t)}$, čia E - tinklo baudos funkcijos reikšmė laiko momentu t.

Kronekerio delta funkcija. Funkcija, kuri palengvina aprašyti tam tikras formules $\delta_{u,v}$, čia funkcija, kuri gražina vienetą, jei kintamieji u ir v sutampa.

Keturi neuroniai tinklai. Visų tinklų veikimo principas yra vienodas, tačiau yra vienas skirtumas. Šie tinklai naudoja skirtingas aktyvacijos funkcijas.

1. **Pirmasis**. Pamiršimo neuroninis tinklas. Žymimas - f. Jis naudoja Sigmoidinę aktyvacijos funkciją.
2. **Antrasis**. Įvesties neuroninis tinklas. Žymimas - i. Jis naudoja Sigmoidinę aktyvacijos funkciją.
3. **Trečiasis**. Įvesties moduliacijos neuroninis tinklas. Žymimas - g. Jis naudoja hiperbolinio tangento aktyvacijos funkciją.
4. **Ketvirtasis**. Išvesties neuroninis tinklas. Žymimas - o. Jis naudoja Sigmoidinę aktyvacijos funkciją.

Naudojamos aktyvacijos funkcijos:

1. **Sigmoidinė** - $f(x) = \frac{1}{1+e^{-x}}$
2. **Hiperbolinio tangento** - $f(x) = tanh(x)$

Bet kurio neuroninio tinklo sandara.

2.2. LSTM išvesties reikšmės apskaičiavimas

Norint apmokyti tinklą ar prognozuoti išvesties reikšmes yra daromas įvesties reikšmių perleidimas per tinklą(angl. *Feed forward*). Pradiniai duomenys ir prieš tai buvusio žingsnio išvesties reikšmės yra paduodamos, kaip įvesties reikšmės į visus keturis rekurentiniame tinkle esamus neuroninius tinklus. Kiekvienas neuroninis tinklas apskaičiuoja išvesties reikšmes pagal atitinkamuose tinkluose esamus svorius ir juose naudojamas aktyvacijos funkcijas.

Pirmo sluoksnio $a_k^{(u,1)}$ ir $h_k^{(u)}$ reikšmės yra pradinės įvesties reikšmės, todėl jos yra paduodamos tiesiogiai. Antro sluoksnio atitinkamo tinklo $a_k^{(u,2)}$ reikšmės yra apskaičiuojamos pagal formulę (1).

$$a_k^{(u,2)} = f(z_k^{(u,2)}), \tag{1}$$

čia funkcija f yra aktyvacijos funkcija, o $z_k^{(u,2)}$ yra apskaičiuojama pagal formulę (2).

$$z_k^{(u,2)} = \sum_{m=1}^{M} w_{mk}^{(u)} h_m^{t-1} + \sum_{m=1}^{I+1} w_{mk}^{(u,1)} a_m^{(u,1)}, \tag{2}$$

čia $z_k^{(u,2)}$ yra suma visų svorių prijungtų prie k-ojo z-o padauginus iš atitinkamų įvesties reikšmių prijungtų prie to svorio.

Toliau trečio ir tolimesnių atitinkamų tinklų sluoksnių neuronų reikšmės $a_k^{(u,l)}$ yra apskaičiuojamos pagal formulę (3).

$$a_k^{(u,l)} = f(z_k^{(u,l)}), \tag{3}$$

čia $z_k^{(u,l)}$ yra apskaičiuojama pagal formulę (4).

$$z_k^{(u,l)} = \sum_{m=1}^{K(u,l-1)+1} w_{mk}^{(u,l-1)} a_m^{(u,l-1)}, \tag{4}$$

čia $z_k^{(u,l)}$ yra suma visų svorių prijungtų prie k-ojo z-o padauginus iš atitinkamų įvesties reikšmių prijungtų prie to svorio.

Apskaičiavus visas atitinkamų tinklų $a_k^{(u,l)}$ reikšmes, kiekvienas neuroninis tinklas gražina paskutinio sluoksnio išvesties reikšmes $a_k^{(u,L)}$.

Toliau yra perskaičiuojama nauja rekurentinio tinklo atmintis, pagal gautas neuroninių tinklų išvesties reikšmes ir prieš tai buvusio žingsnio gražintą atmintį. Nauja atmintis paskaičiuojama pagal formulę (2.2).

$c_k^{(t)} = c_k^{(t-1)} a_k^{(f,L)} + a_k^{(i,L)} a_k^{(g,L)}$, čia $a_k^{(f,L)}$ - pirmojo neuroninio tinklo išvesties reikšmės, $a_k^{(i,L)}$ - antrojo neuroninio tinklo išvesties reikšmės, $a_k^{(g,L)}$ - trečiojo neuroninio tinklo išvesties reikšmės.

Sekantis žingsnis yra apskaičiuoti tarpines išvesties reikšmes $b_k^{(t)}$. Jos yra apskaičiuojamos pagal formulę (5).

$$b_k^{(t)} = a_k^{(o,L)} tanh(c_k^{(t)}), \tag{5}$$

čia $a_k^{(o,L)}$ - ketvirtojo neuroninio tinklo išvesties reikšmės.

Gautoms $b_k^{(t)}$ reikšmėms pritaikome ”softmax” funkciją ir apskaičiuojame tinklo išvesties reikšmes $h_k^{(t)}$ pagal formulę (6). Ši funkcija normalizuoja $b_k^{(t)}$ reikšmes į tikimybinį pasiskirstymą, beto kai kurios reikšmės gali būti neigiamos, tačiau pritaikius šią funkciją reikšmės patenka į intervalą (0,1) ir visų reikšmių suma yra lygi 1.

$$h_k^{(t)} = \frac{e^{b_k^{(t)}}}{\sum_{m=1}^{M} e^{b_m^{(t)}}} \tag{6}$$

2.3. LSTM apmokymas gradientinio nusileidimo metodu

Atliktus įvesties reikšmių perleidimą per tinklą yra skaičiuojamas baudos funkcijos reikšmė, kuri yra lygi vidutinei kvadratinei tinklo paklaidai. Jos formulė yra (7).

$$E^{(t)} = \sum_{k=1}^{M} \frac{1}{2} (y_k^{(t)} - h_k^{(t)})^2, \tag{7}$$

čia $h_k^{(t)}$ - tinklo išvesties reikšmė, o $y_k^{(t)}$ - prognozuojama reikšmė, kurią turime gauti.

Apmokymo tikslas yra mažinti baudos funkcijos reikšmę, tam yra naudojamas gradientinio nusileidimo metodas. Šis metodas leidžia atnaujinti neuroninių tinklų svorius priešinga gradiento kryptimi, kas sumažina vidutinę tinklo išvesties reikšmių paklaidą.

Svorių atnaujinimas vyksta skaičiuojant tinklo baudos funkcijos išvestinę pagal kiekvieną iš tinkle esančių svorių 10. Atnaujinant svorius yra apskaičiuojamas ir saugomas $\Delta w_{ij}^{(u,l)}$. Jis yra apskaičiuojamas naudojant prieš tai buvusį svorio pokytį padauginus iš atitinkamos α reikšmės ir atėmus baudos funkcijos išvestinės reikšmę pagal atitinkamą svorį padaugintą iš η (8). Šie α ir α parametrai nurodo mokymosi greitį. α - nurodo tinklo inertiškumą, tai yra kiek tinklo naujo svorio reikšmę priklauso nuo prieš tai buvusių svorių. η - nurodo tinklo apmokymo greitį, tai yra kiek tinklo nauja svorio reikšmė priklauso nuo dabartiniu laiko momentu įvykdyto apmokymo.

$$\Delta w_{ij}^{(u,l)} = -\eta \frac{\partial E^{(t)}}{\partial w_{ij}^{(u,l)}} + \alpha \Delta w_{ij}^{(u,l)}, \tag{8}$$

čia α - inercija, η - apmokymo greitis.

Tada bendra svorio atnaujinimo gaunama formulė yra (9) :

$$w_{ij}^{(u,l)} = w_{ij}^{(u,l)} + \Delta w_{ij}^{(u,l)} \tag{9}$$

$$\frac{\partial E^{(t)}}{\partial w_{ij}^{(v,s)}} = \sum_{n=1}^{M} \frac{\partial(\sum_{k=1}^{M} \frac{1}{2}(y_m^{(t)} - h_m^{(t)})^2)}{\partial h_n^{(t)}} \frac{\partial h_n^{(t)}}{\partial w_{ij}^{(v,s)}}, \tag{10}$$

čia $\frac{\partial h_k^{(t)}}{\partial w_{ij}^{(v,s)}}$ yra apskaičiuojama pagal formulę (11).

$$\frac{\partial h_k^{(t)}}{\partial w_{ij}^{(v,s)}} = \sum_{n=1}^{M} \frac{\partial(\frac{e^{b_k^{(t)}}}{\sum_{m=1}^{M} e^{b_m^{(t)}}})}{\partial b_n^{(t)}} \frac{\partial b_n^{(t)}}{\partial w_{ij}^{(v,s)}}, \tag{11}$$

čia $\frac{\partial b_n^{(t)}}{\partial w_{ij}^{(v,s)}}$ yra apskaičiuojama pagal formulę (12).

$$\frac{\partial b_n^{(t)}}{\partial w_{ij}^{(v,s)}} = \frac{\partial(a_k^{(o,L)} tanh(c_k^{(t)}))}{\partial w_{ij}^{(v,s)}} = \frac{\partial a_k^{(o,L)}}{\partial w_{ij}^{(v,s)}} tanh(c_k^{(t)}) + a_k^{(o,L)} \frac{\partial tanh(c_k^{(t)})}{\partial w_{ij}^{(v,s)}}, \tag{12}$$

čia išskiriam dvi išvestines $\frac{\partial a_k^{(o,L)}}{\partial w_{ij}^{(v,s)}}$ ir $\frac{\partial tanh(c_k^{(t)})}{\partial w_{ij}^{(v,s)}}$, kurios yra apskaičiuojamos pagal formules (15) ir (13) atitinkamai.

$$\frac{\partial tanh(c_k^{(t)})}{\partial w_{ij}^{(v,s)}} = \frac{\partial tanh(c_k^{(t)})}{\partial c_k^{(t)}} \frac{\partial c_k^{(t)}}{\partial w_{ij}^{(v,s)}}, \tag{13}$$

čia $\frac{\partial c_k^{(t)}}{\partial w_{ij}^{(v,s)}}$ yra apskaičiuojama pagal formulę (14).

$$\begin{aligned} \frac{\partial c_k^{(t)}}{\partial w_{ij}^{(v,s)}} =& \frac{\partial(c_k^{(t-1)} a_k^{(f,L)} + a_k^{(i,L)} a_k^{(g,L)})}{\partial w_{ij}^{(v,s)}} = \\ & \frac{\partial c_k^{(t-1)}}{\partial w_{ij}^{(v,s)}} a_k^{(f,L)} + c_k^{(t-1)} \frac{\partial a_k^{(f,L)}}{\partial w_{ij}^{(v,s)}} + \frac{\partial a_k^{(i,L)}}{\partial w_{ij}^{(v,s)}} a_k^{(g,L)} + a_k^{(i,L)} \frac{\partial a_k^{(g,L)}}{\partial w_{ij}^{(v,s)}} \end{aligned} \tag{14}$$

Toliau išvesime bendrąją $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}}$ formulę, kuria būtų galima apskaičiuoti bet kurio tinklo $a_k^{(u,l)}$ reikšmės išvestinę pagal bet kurį $w_{ij}^{(v,s)}$ (28).

Norint tai atlikti iš pradžių reikia apskaičiuoti bet kurio tinklo paskutiniojo sluoksnio $a_k^{(u,L)}$ išvestinę pagal vienu žemiau esančio sluoksnio svorį $w_{ij}^{(v,L-1)}$ (15).

Pastaba! Tinklų išvesties reikšmės $a_k^{(u,l)}$ yra funkcijos, kurios priklauso nuo visų esančių tinklų svorių. Dėl šitos priežasties yra skaičiuojamos visų tinklų $a_k^{(u,l)}$ išvestinės, pagal visų tinklų svorius.

$$\frac{\partial a_k^{(u,L)}}{\partial w_{ij}^{(v,L-1)}} = \frac{\partial f(z_k^{(u,L)})}{\partial w_{ij}^{(v,L-1)}} = \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \frac{\partial z_k^{(u,L)}}{\partial w_{ij}^{(v,L-1)}}, \tag{15}$$

čia u ir v - nurodo, kad tinklai, kuriose yra kintamieji $a_k^{(u,L)}$ ir $w_{ij}^{(v,L-1)}$ nebūtinai turi sutapti.

Toliau apskaičiuojame $\frac{\partial z_k^{(u,L)}}{\partial w_{ij}^{(v,L-2)}}$ (16).

$$\begin{aligned}\frac{\partial z_k^{(u,L)}}{\partial w_{ij}^{(v,L-1)}} = &\sum_{n=1}^{K(u,L-1)+1} \frac{\partial(\sum_{m=1}^{K(u,L-1)+1} w_{mk}^{(u,L-1)} a_m^{(u,L-1)})}{\partial a_n^{(u,L-1)}} \frac{\partial a_n^{(u,L-1)}}{\partial w_{ij}^{(v,L-1)}} + \\ &\sum_{n=1}^{K(u,L-1)+1} \frac{\partial(\sum_{m=1}^{K(u,L-1)+1} w_{mk}^{(u,L-1)a_m^{(u,L-1)}})}{\partial w_{nk}^{(v,L-1)}} \frac{\partial w_{nk}^{(v,L-1)}}{\partial w_{ij}^{(v,L-1)}} = \\ &\sum_{n=1}^{K(u,L-1)+1} w_{nk}^{(u,L-1)} \frac{\partial a_k^{(u,L-1)}}{\partial w_{ij}^{(v,L-1)}} + \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \delta_{u,v} a_i^{(u,L-1)}\end{aligned} \tag{16}$$

Iš čia gauname, kad sumoje $\sum_{n=1}^{K(u,L-1)+1} \frac{\partial(\sum_{m=1}^{K(u,L-1)+1} w_{mk}^{(u,L-1)a_m^{(u,L-1)}})}{\partial a_n^{(u,L-1)}} \frac{\partial a_n^{(u,L-1)}}{\partial w_{ij}^{(v,L-1)}}$, kai n=K(u,L-1)+1 skaičiuojame Bias neurono išvestinę, pagal svorį. Kadangi Bias neurono reikšmė nepriklauso nuo tinklo svorių, tai $\frac{\partial a_{K(u,L-1)+1}^{(u,L-1)}}{\partial w_{ij}^{(v,L-1)}} = 0$, todėl skaičiuojant šias sumas, neįtrauksime Bias neurono(t.y. n=[1;K(u,L-1)]).

Įstačius išvestinę $\frac{\partial a_n^{(u,L-1)}}{\partial w_{ij}^{(v,L-1)}}$ į formulę (16) ir formulę (16) įstačius į formulę (15) gauname formulę (17).

$$\begin{aligned}\frac{\partial a_k^{(u,L)}}{\partial w_{ij}^{(v,L-1)}} = &\frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \sum_{n=1}^{K(u,L-1)} w_{nk}^{(u,L-1)} \frac{\partial f(z_n^{(u,L-1)})}{\partial z_n^{(u,L-1)}} \\ &\sum_{p=1}^{K(u,L-2)} w_{pn}^{(u,L-2)} \frac{\partial a_p^{(u,L-2)}}{\partial w_{ij}^{(v,L-1)}} + \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \delta_{u,v} a_i^{(u,L-1)}\end{aligned} \tag{17}$$

Pakeičiame skaičiavimų tvarką, taip kad skaičiavimai iš pradžių būtų sumuojami pagal aukštesnio sluoksnio neuronų kiekį, o poto pagal žemesnio. Tada gauname (18) lygybę.

$$\begin{aligned}\frac{\partial a_k^{(u,L)}}{\partial w_{ij}^{(v,L-1)}} = &\sum_{p=1}^{K(u,L-2)} \frac{\partial a_p^{(u,L-2)}}{\partial w_{ij}^{(v,L-1)}} \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \sum_{n=1}^{K(u,L-1)} w_{pn}^{(u,L-2)} w_{nk}^{(u,L-1)} \frac{\partial f(z_n^{(u,L-1)})}{\partial z_n^{(u,L-1)}} + \\ &+ \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \delta_{u,v} a_i^{(u,L-1)}\end{aligned} \tag{18}$$

Šioje lygybėje įsivedame žymėjimą $G_{pk}^{(u,L)} = \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \sum_{n=1}^{K(u,L-1)} w_{pn}^{(u,L-2)} w_{nk}^{(u,L-1)} \frac{\partial f(z_n^{(u,L-1)})}{\partial z_n^{(u,L-1)}}$.

Tuomet gauta nauja lygybė bus (19).

$$\frac{\partial a_k^{(u,L)}}{\partial w_{ij}^{(v,L-1)}} = \sum_{p=1}^{K(u,L-2)} \frac{\partial a_p^{(u,L-2)}}{\partial w_{ij}^{(v,L-1)}} G_{pk}^{(u,L)} + \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \delta_{u,v} a_i^{(u,L-1)} \tag{19}$$

Toliau skaičiuojame $a_k^{(u,L)}$ pagal dviem sluoksniais žemiau esančiais svoriais $w_{ij}^{(v,L-2)}$ (20).

$$\frac{\partial a_k^{(u,L)}}{\partial w_{ij}^{(v,L-2)}} = \frac{\partial f(z_k^{(u,L)})}{\partial w_{ij}^{(v,L-2)}} = \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \frac{\partial z_k^{(u,L)}}{\partial w_{ij}^{(v,L-2)}}, \tag{20}$$

čia u ir v - nurodo, kad tinklai, kuriose yra kintamieji $a_k^{(u,L)}$ ir $w_{ij}^{(v,L-2)}$ nebūtinai turi sutapti.

Toliau apskaičiuojame $\frac{\partial z_k^{(u,L)}}{\partial w_{ij}^{(v,L-2)}}$ (21).

$$\begin{aligned}\frac{\partial z_k^{(u,L)}}{\partial w_{ij}^{(v,L-2)}} = &\sum_{n=1}^{K(u,L-1)} \frac{\partial(\sum_{m=1}^{K(u,L-1)+1} w_{mk}^{(u,L-1)} a_m^{(u,L-1)})}{\partial a_n^{(u,L-1)}} \frac{\partial a_n^{(u,L-1)}}{\partial w_{ij}^{(v,L-2)}} + \\ &\sum_{n=1}^{K(u,L-1)+1} \frac{\partial(\sum_{m=1}^{K(u,L-1)+1} w_{mk}^{(u,L-1)a_m^{(u,L-1)}})}{\partial w_{nk}^{(v,L-1)}} \frac{\partial w_{nk}^{(v,L-1)}}{\partial w_{ij}^{(v,L-2)}} = \\ &\sum_{n=1}^{K(u,L-1)+1} w_{nk}^{(u,L-1)} \frac{\partial a_k^{(u,L-1)}}{\partial w_{ij}^{(v,L-2)}},\end{aligned} \tag{21}$$

čia suma $\sum_{n=1}^{K(u,L-1)+1} \frac{\partial(\sum_{m=1}^{K(u,L-1)+1} w_{mk}^{(u,L-1)a_m^{(u,L-1)}})}{\partial w_{nk}^{(v,L-1)}} \frac{\partial w_{nk}^{(v,L-1)}}{\partial w_{ij}^{(v,L-2)}}$ pasinaikina, nes $\frac{\partial w_{nk}^{(v,L-1)}}{\partial w_{ij}^{(v,L-2)}} = 0$

Taip pat kaip ir prieš tai įstačius išvestinę $\frac{\partial a_n^{(u,L-1)}}{\partial w_{ij}^{(v,L-2)}}$ į formulę (21) ir poto formulę (21) įstačius į formulę (20) gauname formulę (22).

$$\begin{aligned}\frac{\partial a_k^{(u,L)}}{\partial W_{ij}^{(v,L-2)}} = &\frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \sum_{n=1}^{K(u,L-1)} w_{nk}^{(u,L-1)} \frac{\partial f(z_n^{(u,L-1)})}{\partial z_n^{(u,L-1)}} (\sum_{p=1}^{K(u,L-2)} w_{pn}^{(u,L-2)} \frac{\partial a_p^{(u,L-2)}}{\partial w_{ij}^{(v,L-2)}} \\ &+ \sum_{d=1}^{K(u,L-2)+1} \frac{\partial(\sum_{m=1}^{K(u,L-2)+1} w_{mn}^{K(u,L-2)} a_m^{K(u,L-2)})}{\partial w_{dn}^{(u,L-2)}} \frac{w_{dn}^{(u,L-2)}}{\partial w_{ij}^{(v,L-2)}}),\end{aligned} \tag{22}$$

čia

$$\sum_{d=1}^{K(u,L-2)+1} \frac{\partial(\sum_{m=1}^{K(u,L-2)+1} w_{mn}^{K(u,L-2)} a_m^{K(u,L-2)})}{\partial w_{dn}^{(u,L-2)}} \frac{w_{dn}^{(u,L-2)}}{\partial w_{ij}^{(v,L-2)}} = \delta_{u,v} a_i^{(u,L-2)}$$

Toliau iškėlus $\delta_{u,v} a_i^{(u,L-2)}$ iš sandaugos, poto sukeitus skaičiavimų tvarką, kaip ir formulėje (18) gaunname naują lygybę (23) ir taip pat panaikiname iš sumos skaičiavimą su bias neuronais, kaip ir formulėje (17).

$$\begin{aligned}\frac{\partial a_k^{(u,L)}}{\partial w_{ij}^{(v,L-2)}} = &\sum_{p=1}^{K(u,L-2)} (\frac{\partial a_p^{(u,L-2)}}{\partial w_{ij}^{(u,L)}} \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \sum_{n=1}^{K(u,L-1)} (w_{pn}^{(u,L-2)} w_{nk}^{(u,L-1)} \frac{\partial f(z_n^{(u,L-1)})}{\partial z_n^{(u,L-1)}})) + \\ &\delta_{u,v} a_i^{(u,L-2)} \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \sum_{n=1}^{K(u,L-1)+1} w_{nk}^{(u,L-1)} \frac{\partial f(z_n^{(u,L-1)})}{\partial z_n^{(u,L-1)}}\end{aligned} \tag{23}$$

Šioje lygybėje įsivedame žymėjimą $G_{pk}^{(u,L)} = \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \sum_{n=1}^{K(u,L-1)} w_{pn}^{(u,L-2)} w_{nk}^{(u,L-1)} \frac{\partial f(z_n^{(u,L-1)})}{\partial z_n^{(u,L-1)}}$.

Kadangi, gavome taip, kad šitoje lygybėje galime įsivesti tokį patį žymėjimą, kaip ir (19). Tauomet gausime naują formulę (24).

$$\frac{\partial a_k^{(u,L)}}{\partial w_{ij}^{(v,L-2)}} = \sum_{p=1}^{K(u,L-2)} \frac{\partial a_p^{(u,L-2)}}{\partial w_{ij}^{(v,L-2)}} G_{pk}^{(u,L)} + \\ + \delta_{u,v} a_i^{(u,L-2)} \frac{\partial f(z_k^{(u,L)})}{\partial z_k^{(u,L)}} \sum_{n=1}^{K(u,L-1)+1} w_{nk}^{(u,L-1)} \frac{\partial f(z_n^{(u,L-1)})}{\partial z_n^{(u,L-1)}} \tag{24}$$

Turint formules (19) ir (24) galime išvesti bendrą $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}}$ formulę (28).

Išskirsime kelis atvejus, sudarant bendrąją formulę.

1. Kai l = 1, skaičiuojame pirmojo sluoksnio neuronų $a_k^{(u,1)}$ išvestines pagal visus svorius. Pirmąjame sluoksnyje turime du įvesties rinkinius: praeito žingsnio tinklo išvesties reikšmių rinkinys($h_k^{(t-1)}$) ir dabarties įvesties rinkinys($a_k^{(u,1)}$). Kadangi $a_k^{(u,1)}$ rinkinio reikšmės yra konstantos, tai jų išvestinės bus $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}} = 0$. Tuomet pirmojo sluoksnio išvestinės bus lygios $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}} = \frac{\partial h_k^{(t-1)}}{\partial w_{ij}^{(v,s)}}$.

2. Kai l = 2, skaičiuojame antrojo sluoksnio neuronų $a_k^{(u,1)}$ išvestines pagal visus svorius. Antrojo sluoksnio išvestinės yra skaičiuojamos pagal formulę (**??**). Kadangi $a_k^{(u,2)}$ yra apskaičiuojamos pagal formulę (1), tai skaičiuojame šios funkcijos išvestinę (25).

$$\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}} = \frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}} \sum_{n=0}^{M} w_{nk}^{u} \frac{\partial h_n^{(t-1)}}{\partial w_{ij}^{(v,s)}} + \delta_{u,v} \frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}} a_i^{(u,l-1)} \tag{25}$$

3. Kai l >= 3, skaičiuojame trečio ir aukštesnių sluoksnių neuronų $a_k^{(u,1)}$ išvestines pagal visus svorius. Kadangi rekurentinio neuroninio tinklo apmokymui reikia tik kiekvieno neuronio tinklo paskutinio sluoksnio išvestinių pagal visus svorius, todėl kai kurių tarpinių sluoksnių $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}} = \frac{\partial h_k^{(t-1)}}{\partial w_{ij}^{(v,s)}}$ išvestinių nebūtina skaičiuoti.

Iš išvestų formulių (19) ir (24) galime pastebėti, kad skaičiuojant $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}}$ išvestinę, jai reikia jau apskaičiuotos dviem sluoksniais žemesnių neuronų išvestinių pagal tą patį svorį ($\frac{\partial a_k^{(u,l-2)}}{\partial w_{ij}^{(v,s)}}$), todėl galima pastebėti, kad paskutinio sluoksnio $\frac{\partial a_k^{(u,L)}}{\partial w_{ij}^{(v,s)}}$ išvestinė bus apskaičiuojama naudojant jau turimas apskaičiuotas $\frac{\partial a_k^{(u,l-2)}}{\partial w_{ij}^{(v,s)}}$ reikšmes. Skaičiuojant $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}}$ išvestines reikia pastebėti, tai kad svoris pagal kurį yra skaičiuojama išvestinė gali patekti į to pačio tinklo dviejų sluoksnių ribas, pagal kurias skaičiuojame dalines išvestines, todėl reikia išskirti du atvejus: a) kai svoris pagal kurį skaičiuojama $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}}$ yra vienu sluoksniu žemiau(t.y. s+1=l). Esant šiam atvejui iš formulės (19) galime pastebėti, kad prie išvestinės reikšmės turime pridėti $a_i^{(u,l-1)}$, kai sutampa tinklai(u=v) ir svoris pagal kurį yra skaičiuojama išvestinė yra vienu sluoksniu žemiau(s+1=l). Todėl įsivedame delta funkciją, kuri tai realizuoja (26).

$$\delta_{u,v}\delta_{s+1,l} a_i^{(u,l-1)} \tag{26}$$

b) kai svoris pagal kurį skaičiuojama $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}}$ yra dviem sluoksniais žemiau(t.y. s+2=l). Esant šiam atvejui iš formulės (24) galime pastebėti, kad prie išvestinės reikšmės turime pridėti $a_i^{(u,l-1)}$, kai sutampa tinklai(u=v), tačiau kitaip nei atveju a) svoris pagal kurį yra skaičiuojama išvestinė yra dviem sluoksniais žemiau(s+2=l). Todėl įsivedame delta funkciją, kuri tai realizuoja (27).

$$\delta_{u,v}\delta_{s+2,l}a_i^{(u,l-2)}\frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}\sum_{n=1}^{K(u,l-1)+1} w_{nk}^{(u,l-1)}\frac{\partial f(z_n^{(u,l-1)})}{\partial z_n^{(u,l-1)}} \tag{27}$$

Taip pat reikia atkreipti dėmesį, kad pirmojo sluoksnio neuronų išvestinių yra M. Kadangi vektorius K(u,l) nesaugo kiek yra praeito tinklo išvesties reikšmių, tai kai skaičiuosime trečio sluoksnio neuronų išvestines, reikia į tai atsižvelgti(t.y. kai l=3, tai vietoje K(u,1) reikės naudoti M).

Apibendrinus šituos tris punktus gaunama $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}}$ reikšmių skaičiavimo formulė (28).

$$\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v,s)}} = \begin{cases} \text{kai } l=1 & \dfrac{\partial h_k^{(t-1)}}{\partial w_{ij}^{(v,s)}} \\ \text{kai } l=2 & \dfrac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}\displaystyle\sum_{n=0}^{M} w_{nk}^{u}\frac{\partial h_n^{(t-1)}}{\partial w_{ij}^{(v,s)}} + \delta_{u,v}\frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}a_i^{(u,l-1)} \\ \text{kai } l=3 & \displaystyle\sum_{p=1}^{M}\frac{\partial a_p^{(u,l-2)}}{\partial w_{ij}^{(v,s)}}G_{pk}^{(u,l)} + \delta_{l,s+1}\delta_{u,v}\frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}a_i^{(u,l-1)} + \\ & + \delta_{l,s+2}\delta_{u,v}\dfrac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}a_i^{(u,l-1)}\displaystyle\sum_{n=1}^{K(u,l-1)+1} w_{nk}^{(u,l-1)}\frac{\partial f(z_k^{(u,l-1)})}{\partial z_k^{(u,l-1)}} \\ \text{kai } l>3 & \displaystyle\sum_{p=1}^{K(u,l-2)}\frac{\partial a_p^{(u,l-2)}}{\partial w_{ij}^{(v,s)}}G_{pk}^{(u,l)} + \delta_{l,s+1}\delta_{u,v}\frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}a_i^{(u,l-1)} + \\ & + \delta_{l,s+2}\delta_{u,v}\dfrac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}a_i^{(u,l-2)}\displaystyle\sum_{n=1}^{K(u,l-1)+1} w_{nk}^{(u,l-1)}\frac{\partial f(z_k^{(u,l-1)})}{\partial z_k^{(u,l-1)}} \end{cases} \tag{28}$$

Toliau išvesime formulę apskaičiuoti $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v)}}$ reikšmes(atitinkamo neurono išvestinę pagal svorius, kurie jungia praeito žingsnio išvesties reikšmes su esamo žingsnio neuronais) (29).

Šios formules (29) išvedimas yra analogiškas formulės (28) išvedimui, tačiau reikia išskirti esminius skirtumus, kurie supaprastina skaičiavimą.

1. Kai l = 1, formulė nesikeičia, nes išvestinių reikšmės yra paimamos iš praeito žingsnio gautų išvestinių reikšmių. $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v)}} = \frac{\partial h_k^{(t-1)}}{\partial w_{ij}^{(v)}}$

2. Kai l = 2, formulės struktūra nesikeičia, tačiau kai yra skaičiuojama $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v)}}$, tai pridedama $a_i^{(u,l-1)}$ reikšmė yra pakeičiama į $h_i^{(t-1)}$.

3. Kai l = 3, tai skirtingai, nei skaičiuojant formulėje (28), šioje lygybėje dingsta $\delta_{u,v}\frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}h_i^{(t-1)}$,

nes svoriai pagal kuriuos yra skaičiuojama išvestinė priklauso pirmąjam sluoksniui, o kadangi pirmojo sluoksnio išvestines pagal šiuos svorius jau turime, tai $\delta_{l,1}\delta_{u,v}\frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}h_i^{(t-1)}$ $\sum_{n=1}^{K(u,l-1)+1} w_{nk}^{(u,l-1)}\frac{\partial f(z_k^{(u,l-1)})}{\partial z_k^{(u,l-1)}}$ ši dalis išlieka išskyrus kaip ir antruoju atveju $a_i^{(u,l-1)}$ reikšmė yra pakeičiama į $h_i^{(t-1)}$.

4. Kai l > 3, tai abi sumos paminėtos antruoju ir trečiuoju atveju dingsta, nes skaičiuojant trečio ir aukštesnio sluoksnio neuronų išvestines, svoriai pagal kuriuos yra skaičiuojamos išvestinės yra trijais ir daugiau sluoksnių žemiau, kadangi formulė pagal kurią skaičiuojame išvestines rekurentiškai naudoja jau apskaičiuotas dviem sluoksniais žemiau esančių neuronų išvestines, todėl paminėtos sumos yra nereikalingos.

Atlikus šiuos pakeitimus yra gaunama $\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v)}}$ reikšmių skaičiavimo formulė (29).

$$\frac{\partial a_k^{(u,l)}}{\partial w_{ij}^{(v)}} = \begin{cases} \text{kai } l = 1 & \dfrac{\partial h_k^{(t-1)}}{\partial w_{ij}^{(v)}} \\ \text{kai } l = 2 & \dfrac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}} \displaystyle\sum_{n=0}^{M} w_{nk}^{u} \frac{\partial h_n^{(t-1)}}{\partial w_{ij}^{(v)}} + \delta_{u,v}\frac{f(z_k^{(u,l)})}{z_k^{(u,l)}}h_i^{(t-1)} \\ \text{kai } l = 3 & \displaystyle\sum_{p=1}^{M} \frac{\partial a_p^{(u,l-2)}}{\partial w_{ij}^{(v)}} G_{pk}^{(u,l)} + \delta_{l,1}\delta_{u,v}\frac{\partial f(z_k^{(u,l)})}{\partial z_k^{(u,l)}}h_i^{(t-1)} \sum_{n=1}^{K(u,l-1)+1} w_{nk}^{(u,l-1)}\frac{\partial f(z_k^{(u,l-1)})}{\partial z_k^{(u,l-1)}} \\ \text{kai } l > 3 & \displaystyle\sum_{p=1}^{K(u,l-2)} \frac{\partial a_p^{(u,l-2)}}{\partial w_{ij}^{(v)}} G_{pk}^{(u,l)} \end{cases} \tag{29}$$

2.4. LSTM taikymas

Šis sudarytas rekurentinis neuroninis tinklas bus taikomas kalbos modeliavime. Tiksliau tinklas bus apmokinamas prognozuoti vartotojo rašomo žodžio pabaigą ir pasiūlyti sekantį žodį (angl. *Query auto-complete*).

Duomenys

Tinklui apmokinti bus naudojama internete rasta bet kokia knyga. Šios knygos tekstas bus suskaidytas į atskirus sakinius, kurie bus naudojami, kaip atskiri apmokymo rinkiniai.

Apmokymas

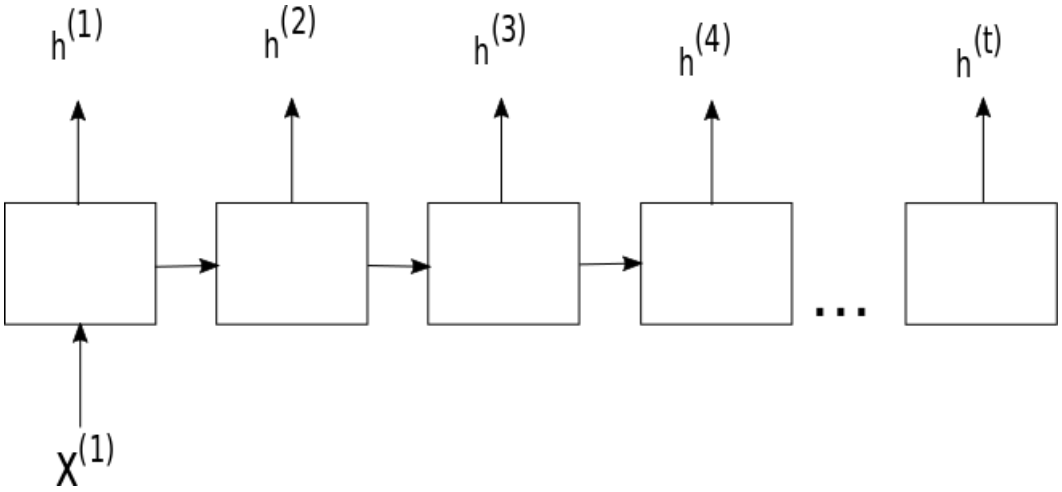
Kiekvienoje apmokymo iteracijoje bus paduodamas vienas sakinys. Šis sakinys suskaidytas į simbolių masyvą, kur kiekvienas simbolis bus traktuojamas, kaip viena įvesties reikšmė. Tinklo apmokymas vyksta, kai yra paduodama tam tikra simbolių seka ir pagal ją yra prognozuojamas tolimesnis tekstas. Didėjant įvesties simbolių sekai iš duotojo sakinio, prognozuojamas tolimesnis tekstas turėtų artėti arba būti panašus į esamą paduotą sakinį.

Pasirinkta tinklo apmokymo strategija:

1. Iš pradžių bus paduota pirma sakinio raidė, kaip įvestis. Ši įvestis bus praleidžiama pro tinklą ir tinklas prognozuos sekančia raidę. Tuomet atliksime tinklo apmokymą, pagal prognozuojamą raidę ir ją lyginsime su duotojo sakinio antrąją raide. Tuomet atlikę apmokymą, pagal prognozuotą raidę prognozuosime sekančią sakinio raidę. Taip ciklą kartosime tol kol įvyks vienas iš dviejų variantų:

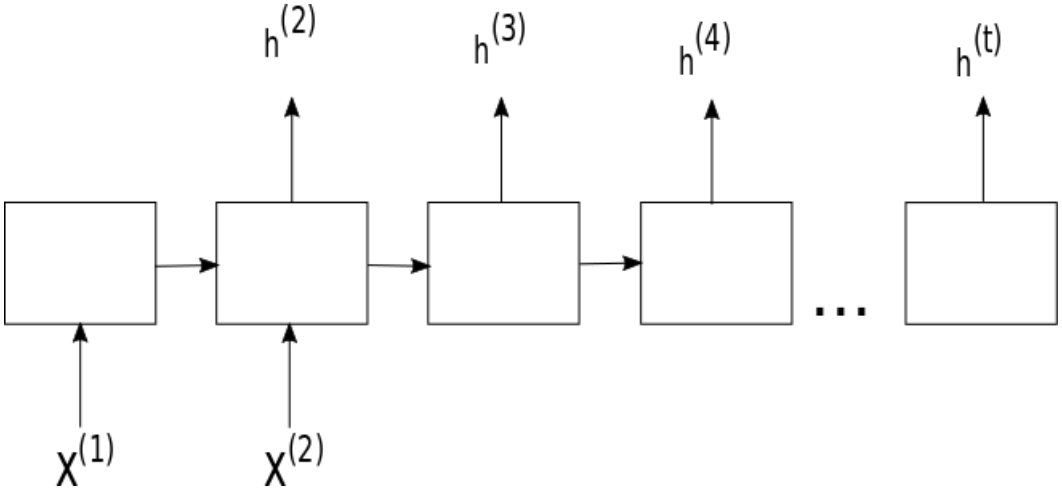
   (a) jau būsime prognozavę pirmus du sakinio žodžius

   (b) pasieksime sakinio pabaigą ir nebegalėsime atlikti apmokymo



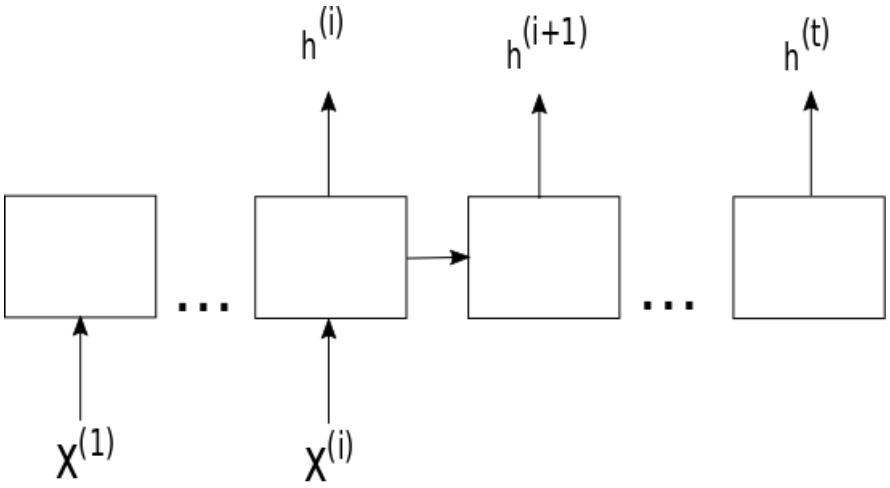
14 pav. Tinklo apmokymas, kai paduodama viena sakinio raidė.

2. Toliau bus paduodamos dvi pirmos sakinio raidės, kaip įvestis. Pagal šias įvestis bus prognozuojama trečioji sakinio raidė ir tinklas bus apmokomas lyginant šią raidę su duotojo sakinio trečiaja raide. Šį ciklą taip pat kartosime tol kol įvyks vienas iš dviejų variantų paminėtų pirmame punkte.



15 pav. Tinklo apmokymas, kai paduodamos dvi sakinio raidės.

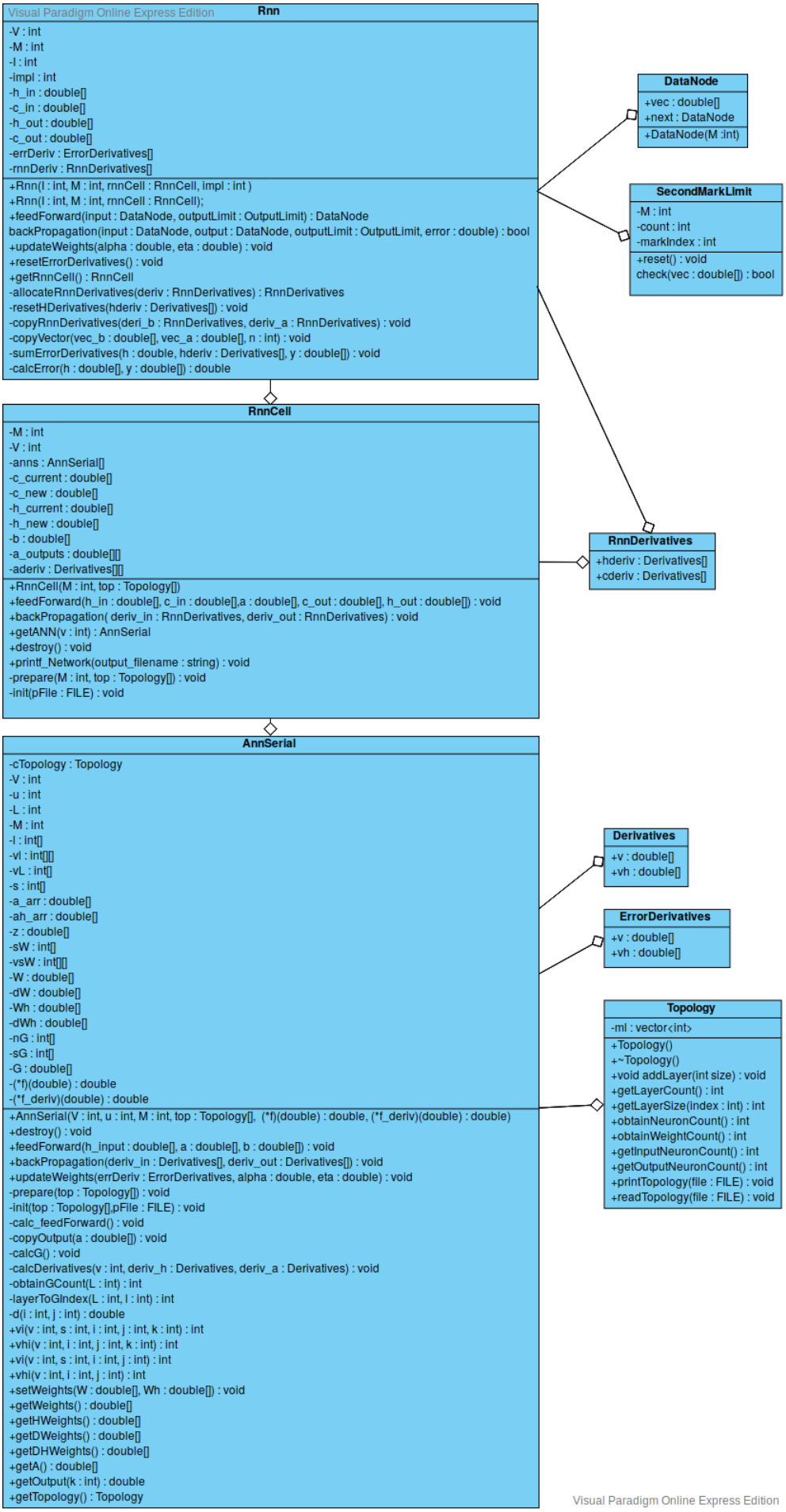
3. Tokiu būdu didinsime paduodamų sakinio raidžių kiekį, kaip įvestis. Pagal šias įvestis prognozuosime sekančią sakinio raidę, kurią lyginant su atitinkama duotojo sakinio raide apmokinsime tinklą. Šį ciklą kartosime iki tol kol įvyks vienas iš dviejų variantų paminėtu prieš tai buvusiuose punktuose arba paduodamų sakinio raidžių kiekis sutaps su duotuoju sakiniu.



16 pav. Tinklo apmokymas, kai paduodamos kelios sakinio raidės.

2.5. Programinė įranga

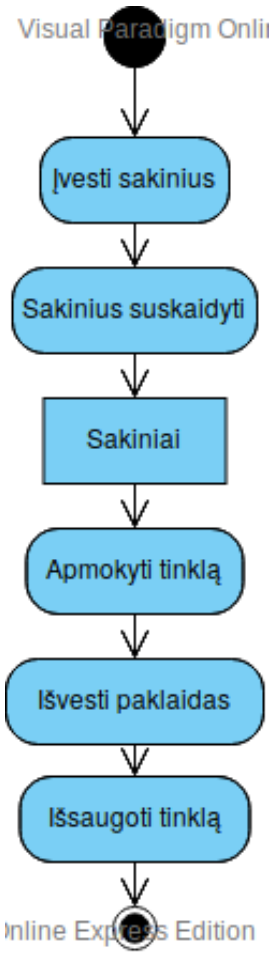
Darbe kurtas rekurentinis neuroninis tinklas buvo realizuotas C++ programavimo kalba. Naudojama operacinė sistema yra Ubuntu, kuri yra sukurta Linux pagrindu. Projektas buvo saugomas gitHub repozitorijoje.



17 pav. Klasiu diagrama

Tinklui realizuoti buvo sukurtos klasės, kurios aprašo tinklo veikimą.

1. **Rnn** - klasė, kuri kontroliuoja viso tinklo veikimą. Apdoroja įvesties duomenis, apskaičiuoja prognozuojamas reikšmes, tinklą apmokina.
2. **RnnCell** - klasė, kuri kontroliuoja rekurentinio neuroninio tinklo skaičiavimus susijusius su viena apmokymo ar prognozavimo iteracija.
3. **AnnSerial** - klasė, kuri kontroliuoja paprastų neuroninų tinklų skaičiavimus susijusius su viena apmokymo ar prognozavimo iteracija.
4. **Topology** - klasė, kuri saugo atitinkamo neuroninio tinklo struktūrą.
5. **Derivatives** - klasė. kuri saugo atitinkamo tinklo išvesčių išvestines, pagal atitinkamo neuroninio tinklo svorius.
6. **ErrorDerivatives** - klasė. kuri saugo viso tinklo išvesčių išvestines, pagal atitinkamo neuroninio tinklo svorius.
7. **RnnDerivatives** - klasė, kuri saugo visų tinklų išvestines, pagal visų tinklų svorius.
8. **SecondMarkLimit** - klasė, kuri skaičiuoja kiek kartų atitinkamas simbolis buvo panaudotas, pagal kurį yra prognozuojamas tekstas.



18 pav. Veiklos diagrama.

18 paveiksle pavaizduota tinklo apmokymo veiklos diagrama. Iš pradžių yra paduodami duomenys į tinklą, šie duomenys yra suskaidomi į sąkinius. Tinklas apmokomas tinklą apmokinant kas vieną sakinį ir grąžinant to sakinio apmokymo vidutinės paklaidos reikymę. Kai yra apmokomi visi sakiniai tinklas gražina kiek laiko mokinosi ir išsisaugo apmokinto tinklo parametrus, kad jį būtų galima naudoti ateityje prognozuojant reikšmes.

3. Eksperimentiniai skaičiavimai

Šioje dalyje apmokant tinklą analizuosiu tinklo apmokymą apibūdinančius parametrus:

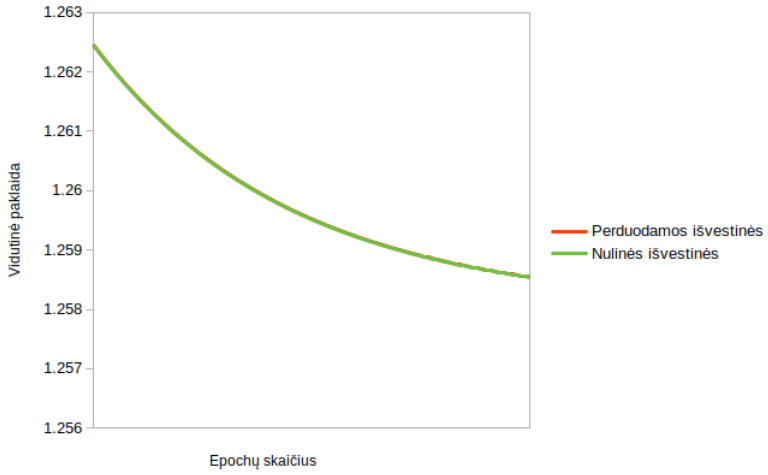
1. Palyginsiu tinklo apmokymo vidutines paklaidas, kai tinklo naujam apmokymui yra naudojamos praeito apmokymo dalinės išvestinės su tinklo apmokymu, kai dalinės išvestinės po kiekvienos apmokymo iteracijos yra nunulinamos.
2. Lyginsiu tinklo vidutines paklaidas priklausant nuo apmokymų skačiaus su skirtingomis α ir η reikšmėms.
3. Atliksiu tinklo apmokymo greitaveikos tyrimą, esant skirtingoms tinklo topologijoms.

Tinklo apmokymo vidutinių paklaidų lyginimas, kai naujam tinklo apmokymui yra naudojamos praeito žingsnio dalinės išvestinės su tinklo apmokymu, kai dalinės išvestinės po kiekvieno apmokymo yra nunulinamos.

Šis tyrimas bus atliekamas prie vienodų tinklo parametrų. $\alpha = 0.8$ $\eta = 0.2$ Topologija - įvesties h reikšmių bus M, įvesties x reikšmių bus I+1, išvesties reikšmių, taip pat bus M.

Abiejais atvejais bus vykdomas apmokymas atliekant n=100 epochų.

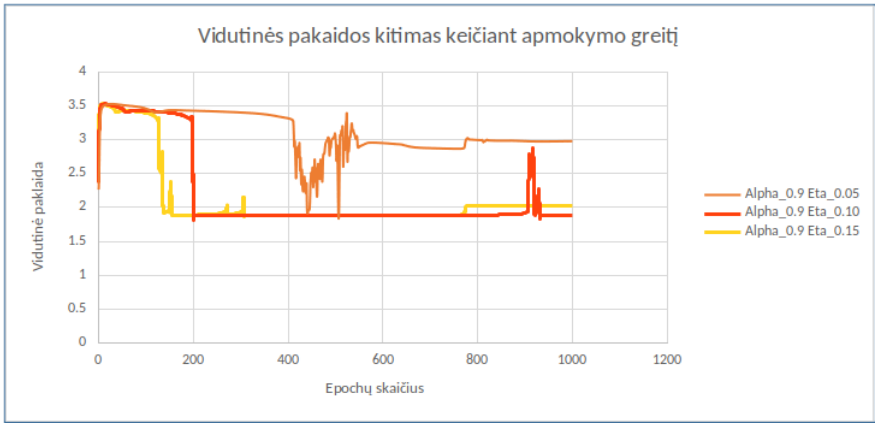
Lyginsime tinklo vidutines paklaidas po kiekvienos apmokymo epochos. (19)



19 pav. Apmokymo paklaidos.

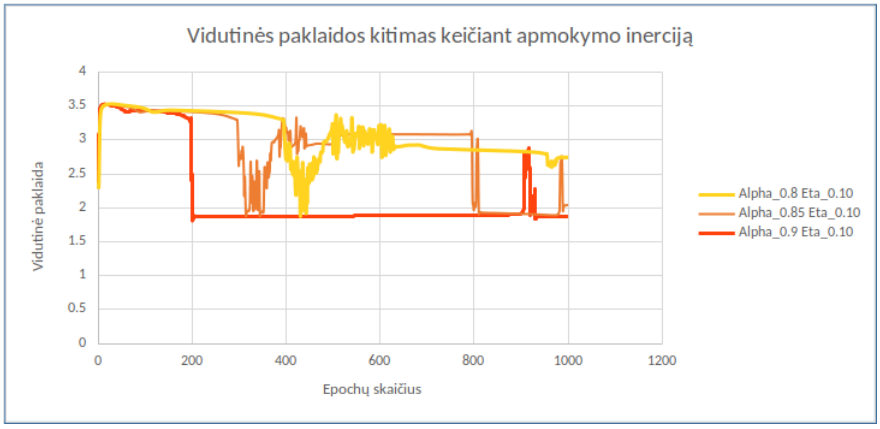
Tinklą apmokant naudojant abu būdus, tinklo apmokymo klaidos funkcijų grafikai yra identiški, todėl nėra būtina šių paklaidų perduoti atliekant naują apmokymą.

Lyginsiu tinklo vidutines paklaidas priklausant nuo apmokymų skačiaus su skirtingomis α ir η reikšmėms.



20 pav. Tinklo apmokymo paklaidos grafikas su skirtingais apmokymo greičiais.

Iš grafiko 20 galima pastebėti, kad tinklas geriausiai apsimoko, kai apmokymo greičio koeficientas $\eta = 0.15$, tačiau grafikas gaunasi tiesinis po 200-os epochos, tai galėjo nutikti dėl to, kad funkcija pateko į lokalaus minimumo tašką. Taip pat tinklas panašiai apsimoko, kai apmokymo greičio koeficientas yra $\eta = 0.1$, tačiau galima pastebėti, kad ties 950-a epocha yra kažkokių nukrypimų. Kai $\eta = 0.05$, tai tinklas apsimoko prasčiausiai, intervale (400;600) grafike taip pat galima pastebėti nukrypimų, tačiau nuo 600-os epochos tinklo vidutinė paklaida ima tolygiai mažėti.



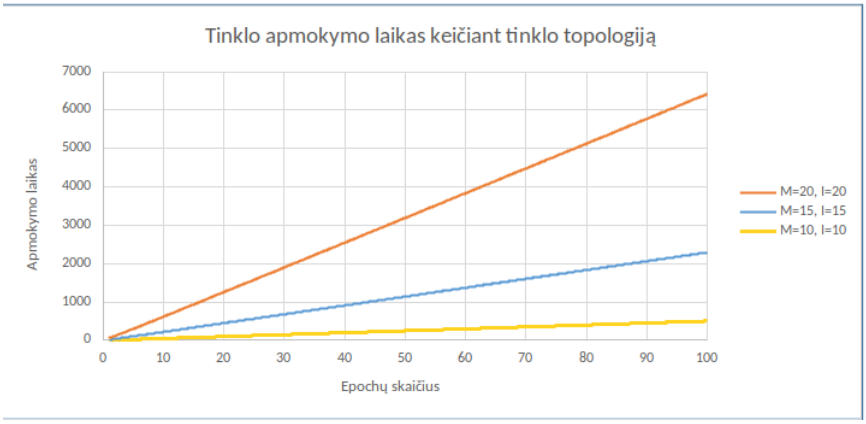
21 pav. Tinklo apmokymo paklaidos grafikas su skirtingom apmokymo inercijom.

Iš grafiko 21 galima pastebėti, kad tinklas geriausiai apsimoko, kai apmokymo inercijos koeficientas $\alpha = 0.9$, tačiau grafikas gaunasi tiesinis po 200-os epochos, tai galėjo nutikti dėl to, kad funkcija pateko į lokalaus minimumo tašką. Tinklas apsimoko panašiai iki 800-os epochos prie koeficientų $\alpha = 0.85$ ir $\alpha = 0.8$, o ties 400-a epocha yra kažkokių nukrypimų, tačiau nuo 800-os epochos grafikai išsiskiria ir kai $\alpha = 0.85$ tinklo apmokymas susilygina su tinklo apmokymu, kai $\alpha = 0.9$.

Atliksiu tinklo apmokymo greitaveikos tyrimą, esant skirtingoms tinklo topologijoms.

Iš grafiko galima pastebėti, kad vidutiniai vienos epochos apmokymo laikai, kai tinklo topologijos yra

1. M=10, I=10 (840 svorių) - 5 sekundės.
2. M=15, I=15 (1860 svorių) - 23 sekundės.
3. M=20, I=20 (3280 svorių) - 64 sekundės.



22 pav. Tinklo apmokymo laiko palyginimas keičiant tinklo topologijas.

Galima pastebėti, kad didinant tinkle esančių svorių kiekį, tinklo apmokymo laikas didėja eksponentiškai.

4. Išvados

1. Buvo sudarytos ir išvestos LSTM rekurentinio neuroninio tinklo formulės. Pavyko išvesti bendrąsias tinklo apmokymo formules gradientinio nusileidimo metodu. Šios formulės buvo optimizuotos, naudojant algoritmų analizės dinaminio programavimo metodą, kai tinklo išvesčių reikšmių išvestinės apskaičiuojamos nenaudojant rekurencijos ”top-down” metodika, o naudojama ”bottom-up” metodika, kai iš pradžių yra apskaičiuojamos tarpinės reikšmės, jos išsaugomos ir apjungiamos.

2. Tinklas buvo realizuotas programiškai. Tinklas implementuotas taip, kad būtų galima patogiai keisti įvesties ir išvesties vektorių ilgius, būtų galima pasirinkti kiekvieno viduje esančio neuroninio tinklo individualias topologijas, nustatyti šių tinklų naudojamas aktyvacijos funkcijas ir pasirinkti apmokymui naudojamų parametrų reikšmes.

3. Tinklo atskiri metodai buvo ištestuoti ir metodai veikia korektiškai.

4. Tinklas yra pritaikytas apmokymui duotuoju tekstu ir tada tinklas išsaugomas faile. Šį tinklą poto galima pasileisti ir prognozuoti sakinius, tačiau tinklui korektiškai prognozuoti tekstą reikia atlikti labai daug apmokymų.

5. (a) Apmokant tinklą nėra būtina perduoti naujai apskaičiuotų išvestinių į tolimesnių žingsnių apmokymus, kadangi tinklas yra apmokomas vienodai abiejais atvejais. Remiantis šiuo faktu galima supaprastinti išvestas formules, kurias realizavus tinklo apmokymo laikas būtų daug trumpesnis.

   (b) Naudojant skirtingus apmokymo parametrus tinklas apsimokina skirtingai, tačiau visus atvejus sieja, kad apmokant šį tinklą naudojant bet kokius parametrus, tinklo funkcija patenka į lokalaus minimumo tašką, dėl kurio tinklas pilnai neapsimoko.

   (c) Tinklo apmokymas, kai įvesties reikšmių vektorių ir prognozuojamų reikšmių vektorių ilgiai didėja, tai tinklo apmokymo trukmė didėja eksponentiškai, todėl apmokant tinklą įvesties reikšmių parametrus būtų galima klasifikuoti į klases, kurios leistų tinklui greičiau apsimokinti.

Literatūros sąrašas

1. SUKHADEVE, A. *Understanding Neural Network: A beginner's guide*. Ashish Sukhadeve, 2017.
2. TCH, A. The mostly complete chart of neural networks. 2017. doi:10.1002/aic.690381003.
3. IGNATAVIčIENė, I. *Feedforward neural network systems a comparative analysis*. Ieva Ignatavičienė, 2012.
4. V, A. S. Understanding activation functions in neural networks. 2017.
5. DAS, S. Feed-forward neural networks with mxnetr. 2017.
6. MIRJALILI, S. *Evolutionary Algorithms and Neural Networks. Theory and Applications*. Springer, 2019. ISBN 978-3-319-93025-1.
7. DEIVIDAS RIABčINSKIS, T. M. Neuroninių tinklų apmokymas atpažinti ranka rašytinius skaičius, naudojant cuda biblioteką. 2018.
8. HU, X. and P. BALASUBRAMANIAM. *Recurrent Neural Networks*. InTech, 2008. ISBN 978-953-7619-08-4.
9. FILIPPO MARIA BIANCHI, M. C. K. A. R. R. J., Enrico Maiorino. *Recurrent Neural Networks for Short-Term Load Forecasting*. Springer, 2017. ISBN 978-3-319-70338-1.
10. OLAH, C. Understanding lstm networks. 2015.
11. WANG, C.-F. The vanishing gradient problem. 2019.
12. ADRIAN J. SHEPHERD BA, P. a., MSc. *Second-Order Methods for Neural Networks: Fast and Reliable Training Methods for Multi-Layer Perceptrons*. 1st edition. Springer-Verlag London, 1997. ISBN 978-3-540-76100-6,978-1-4471-0953-2.
13. PARK, D. H. and R. CHIBA. A neural language model for query auto-completion. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2017, pp. 1189–1192. ISBN 978-1-4503-5022-8. doi:10.1145/3077136.3080758.
    URL `http://doi.acm.org/10.1145/3077136.3080758`
14. RUSH D. ROBINETT III, G. R. E. J. E. H., David G. Wilson. *Applied dynamic programming for optimization of dynamical systems*. Society for Industrial and Applied Mathematics, 2005. ISBN 9780898715866,0898715865.
15. FAZLUL KADER MURSHED NAWAZ, K. G. J. G. D. M. R. N. S., Arnab Chattopadhyay. Comparison of open mp and cuda. 2014.