TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Jaan Erik Pihel

Modulaarsed tehisnärvivõrgud

Bakalaureusetöö (9 EAP)

Juhendaja: Meelis Kull, PhD

Juhendaja: Markus Kängsepp, MSc

Tartu 2019

Modulaarsed tehisnärvivõrgud

Lühikokkuvõte: Töös uuriti modulaarsete tehisnärvivõrkude oskust õppida lahendama probleeme, mis on olemuslikult alamülesanneteks jaotatavad. Ülesandeks valiti närvivõrgu abil kaadri kaupa palli liikumise ennustamine laual, kus algsisenditeks on palli ja seinte koordinaadid koos kiirustega ning kus toimub sirgliikumine ning põrked vastu kõiki seinu. Töös loodud närvivõrgu arhitektuur, kus on iga kihti võimalik loomulikult tõlgendada, kasutab tähelepanumoodulit, et liigitada alamülesandeid. Näidatakse, et sellise mudeliga õppimine on ilma moodulite eeltreenimiseta küllalt aeglane. Kuid eeltreenimisel võib mudel õppida ülesandeid lahendama vähesema ülesobitusega, sest mudeli arhitektuuri on teatud eeldusi ja inimteadmisi sisse kodeeritud.

Võtmesõnad:
Modulaarne närvivõrk, eelduste tegemine närvivõrgus, kaofunktsiooni kujutamine.

CERCS: P176

Modular neural networks

Abstract: The work focused on the ability of modular artificial neural networks to solve problems which are inherently divisible into smaller subtasks. The chosen problem was to use neural networks in predicting frames of ball movements and bounces on a table with borders. The network had ball and border coordinates and ball speed as inputs. A neural architecture was developed where each layer is easily interpreted and which uses attention to classify tasks. It is shown that without pre-training the learning is quite slow. However, with pre-training, the model can overfit less than different networks, because the model architecture has some bias and human knowledge coded into it.

Keywords:
Modular artificial neural net, biases in neural networks, depiction of loss landscapes.

CERCS: P176

Sisukord

Sissejuhatus

Viimasel ajal on neurovõrke kasutavad masinõppe meetodid saanud üha paremaid tulemusi erinevate klassifitseerimise, keeletöötluse ning mängude mängimisega seotud ülesannete lahendamisel [21]. Paremaid tulemusi põhjendatakse osalt sellega, et arvutuskiirus on tublisti tõusnud [19], sest graafikakaardid on läinud paremaks ja odavamaks ning neil osatakse neurovõrke treenida paralleelselt [3]. Teisalt, on jõutud paremate algoritmideni ning saadakse aru, millises suunas konkreetsed algoritmid või neurovõrguarhitektuurid kallutavad tulemust.

Masinõppe ja neuroteaduse alad on praegu lähedalt seotud – tihti saadakse ideid uuteks arhitektuurideks inimajult. Konvolutsiooniliste võrkude idee on sarnane sellele, kuidas inimese nägemissignaale töötlevad närvirakud on struktureeritud. Pertseptroni saab võrrelda inimese ajus olevate neuronitega. Rekurrentsed võrgud proovivad simuleerida seda, kui sama kiht inimajus saab ajas erinevat informatsiooni, sh iseenda eelmise ajahetke väljundit.

On uuritud, kuidas inimaju õpib abstraktselt midagi uut [16] ning kuidas inimene võiks seostada uue probleemiga tutvudes juba varasemalt õpitut [1]. Näiteks on välja toodud, et inimene seostab ümbritsevat iseenda kehaga: seepärast on keeles sellised mõisted nagu lauajalg, peamine, põhjanaba jne. Algoritm, mis meie aju arenemist ja õppimist perfektselt simuleeriks, oskaks väga efektiivselt luua analoogiad, sest inimesel on hea abstraheerimisvõime.

Süvamõtlemine toimub ajus prefrontaalses korteksis, mis koosneb minikolonnidest, mille sisendiks ja väljundiks on samasugusel kujul informatsioon [5]. Need koosnevad 80-120 neuronist ja hinnanguliselt on neid ajus 200 miljonit. Ray Kurzweil on raamatus „How to Create a Mind“ [16] välja toonud teooria, kuidas II, III, IV ja V kihi kolonnid omavahel ühendatud on. Näiteks kui III kiht tunneb ära sõnu ja II kiht kirjatähti, siis kui üks täht on puudu, aitab III kiht seda pakkuda. Sellist taseme kaupa probleemi abstrahheerimist on keeruline saavutada iga närvivõrgu kihiga, kuid võiks olla saavutatav

modulaarse arhitektuuriga.

Töös luuakse füüsikaline simulatsioon, milles kujutatakse pallide liikumist ja põrkamist laual. Bakalaureusetöö eesmärk on ennustada selles simulatsioonis modulaarse närvivõrgu abil palli liikumist järgnevatel kaadritel ning sealjuures selgitada, miks modulaarne õppimine on kasulik.

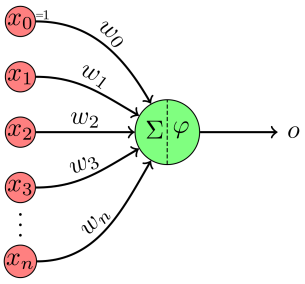
1 Taust

1.1 Tehisnärvivõrgud ja tagasilevi

Edasilevi võrgud ehk mitmekihilised pertseptronid (*MLP*) on mudelid, mille eesmärgiks on lähendada funktsioone [7]. Selleks on mudelil kasutada k kihti funktsioone $f^{(1)}, f^{(2)}, ..., f^{(k)}$. Juhendajaga õppides suudetakse täpsemalt väljendada funktsiooni, mille põhjal treeningandmestik on loodud. Kui treeningandmestik paaridega $(\mathbf{x}, \mathbf{y})$ (siin kirjeldatakse tumedas kirjas muutujaid vektorite või maatriksitena) on loodud funktsiooniga f^*, kus $f^*(\mathbf{x}) = \mathbf{y}$, siis pärast edukat õppimist kehtib $f^*(\mathbf{x}) \approx f(\mathbf{x}) = f^{(k)}(...f^{(2)}(f^{(1)}(\mathbf{x})))$, kus tehte $f(\mathbf{x})$ sooritamist kutsutakse edasileviks. Iga kiht $f^{(i)}$ koosneb neuronitest ning viimase kihi neuroneid loetakse väljundneuroniteks ning sellele eelnevaid peitneuroniteks. Igal neuronil on summeerimisfunktsioon $\Sigma(\mathbf{x}, \theta)$ ja aktivatsioonifunktsioon ϕ (joonis 1) ning $f^{(i)} = \phi(\Sigma(\mathbf{x}, \mathbf{w}) = \phi(\sum_i \mathbf{w}_i \mathbf{x}_i))$. Neuroni väljund on sisendiks kõikidele järgmise kihi neuronitele.

Õpitavateks parameetriteks on närvivõrgu puhul kaalud θ. Kirjeldagu mudelit funktsioon $f(\mathbf{x}, \theta)$, kus iga kihi i jaoks leidub kaalumaatriks θ_i ning j-ndat i kihi neuronit ja k-ndat $i-1$ kihi neuronit ühendab kaal θ_{ijk}. Sarnaselt teistele töödele lisatakse ka siin kohale k=0 igale kihile juurde üks neuron, mille väljund on alati 1 ning mida loetakse vabaliikmeks. Iga j-nda neuroni sisendiks on terve eelmise kihi väljund $\mathbf{x}$. Summeerimisfunktsioon on $h_j = \sum_{k=0}^{m} x_k \theta_{ijk}$, kus m on eelneva kihi neuronite arv. Seejärel rakendatakse aktivatsioonifunktsiooni $o_j = \phi(h_j)$, kus aktivatsioon võib olla lineaarne (nt *ReLU*) või mittelineaarne (nt sigmoid, hüperboolne tangens).

Lineaarne regressioon on üheneuroniline lineaarse aktivatsiooniga $\phi(x) = x$ närvivõrk ning seda on võimalik lahendada analüütiliselt, kus $\mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$ ning treeningolemid on $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, \ldots)$ ja -märgendid $\mathbf{y} = (y_1, y_2, \ldots)$.



Joonis 1. Pertseptroni arhitektuur, kus $x_0 = 1$ ja $w_0 = b$ [22].

Nii logistilise regressiooni, kus aktivatsioonifunktsioon on $\phi(x) = \frac{1}{1+e^{-x}}$, kui närvivõrkude puhul toimub õppimine numbriliste meetodite abil. Enim kasutatakse selleks gradientlaskumist. Gradientlaskumiseks arvutatakse viga ning seejärel kihi kaupa sisendite ja kaalude järgi tuletise võtmisest teatakse, et kui palju ja mis suunas muutub viga väikse kaalumuudatuse tagajärjel. Gradiendist on teada, mis suunas maksimeerida kaotust ning sellele vastupidises suunas on võimalik teha õppesammu suurusest lähtuvalt väike kaaluparandus.

Näitlikustamaks gradientlaskumist võib vaadelda viimase peitkihi kaaluparandusi vähemalt 2 kihilise närvivõrgu puhul. Olgu väljundkihil 1 neuron, kihil enne seda n neuronit ning $\mathbf{w}$ neid ühendav kaaluvektor. Olgu ühe treeningolemi edasilevil eelviimase kihi väljundvektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ ning viimase kihi väljund $\hat{y}$ ning Λ mingi kaofunktsioon, mis hindab tehtud viga igal väljundneuronil $\Lambda(y, \hat{y}) = L$, kus y on õige oodatav vastus. Et leida $\frac{\partial L}{\partial x}$ ja $\frac{\partial L}{\partial W}$, piisab ahelareegli leida $\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{x}}$ ja $\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{w}}$, kus

$$\frac{\partial \hat{y}}{\partial \mathbf{x}} = \left(\frac{\partial \hat{y}}{\partial x_1}, \ldots, \frac{\partial \hat{y}}{\partial x_n} \right) \text{ ja} \qquad \frac{\partial \hat{y}}{\partial \mathbf{w}} = \left(\frac{\partial \hat{y}}{\partial w_0}, \frac{\partial \hat{y}}{\partial w_1}, \ldots, \frac{\partial \hat{y}}{\partial w_n} \right).$$

Nii on vektorargumentide puhul sooritatud tehted

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial \mathbf{x}} \tag{1.1}$$

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial \mathbf{w}}, \tag{1.2}$$

kus valem 1.1 tähistab osatuletist, mida kantakse eelnevatele kihtidele edasi ning valem 1.2 osatuletist, millega sooritatakse kaaluparandus $W_{uus} = W_{vana} - \lambda \frac{\partial L}{\partial W}$, kus λ on õpisamm. Õpisamm on väike, et tehtel $\frac{df}{dx} = \frac{f(x+\lambda)-f(x)}{\lambda} + \varepsilon$ ei läheks viga ε suureks. Viga on võimalik hinnata Taylori reaga $\varepsilon = \frac{d^2f}{dx^2}(x+\lambda)\frac{1}{2!}\lambda + \frac{d^3f}{dx^3}(x+\lambda)\frac{1}{3!}\lambda + \ldots$. Juhul kui kaofunktsioon on piisavalt sujuv, siis $\frac{df}{dx}(x+\lambda)$ annab lähedasema tulemuse kui mittesujuva puhul ning võib kasutada suuremat õpisammu.

1.2 Eeldused närvivõrgu arhitektuurides

Eelmises peatükis kirjeldati närvivõrkude täielikult ühendatud kihte ning probleem, mis nendega tekkida võib, on mürast tulenevate juhuslike seoste tegemine. Sellise arhitektuuri puhul õpivad mudelid sageli treeningandmetest õpitud vajalikke tulemusi pakkuma valedel põhjustel: näiteks ei soovita, et pildil olevat kassi ennustataks selle järgi, et kõige vasapoolsem alumine piksel on roheline. Selline nähtus on tuntud kui ülesobitamine (*overfitting*). Kui treeningandmeid oleks piisavalt ning õppimiseks kuluv arvutuskulu ei oleks segav asjaolu, võiks selline võrk lõpuks saada aru, millised ühendused närvivõrgus on päriselt olulised ning millised mitte. Samuti on loodud eelnevalt kirjeldatud probleemiga tegelemiseks lihtsamaid mehhanisme nagu väljajätumeetod [10] ja regularisatsioon [8].

Küll aga on kiirem ja odavam arhitektuuri teatud eeldusi sisse arvestada. Pildi klassifitseerimisülesannetel kasutatakse konvolutsioonilisi närvivõrke, mille arhitektuur mõjutab järgnevaid omadusi [13]:

- konvolutsiooni filtrid on ühendatud vaid L2 kauguse järgi lähimate tunnustega, et uusi tunnuseid arvutada,

- konvolutsiooniliste filtrite kasutamine mõjutab seda, et mudeli esimestel kihtidel ei ole tunnuse jaoks oluline, kas see asetseb näiteks pildi all- või ülaosas.

Rekurrentsete võrkude puhul tehakse eeldus, et eelnevate ajahetkede väljundid mõjutavad tulemust. Selliste eelduste tegemine aitab muuhulgas aegridade ja keeletehnoloogiaga seotud ülesandeid lahendada.

1.3 Modulaarsed närvivõrgud varasemas kirjanduses

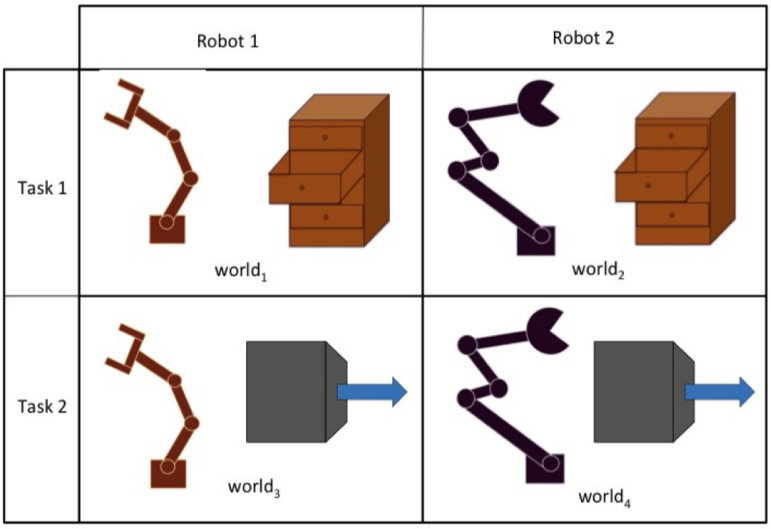
Modulaarsed närvivõrgu arhitektuurid võimaldavad inimteadmisi ja eeldusi ülesannete lahendamisel sisse arvestada. Nii on võimalik kirjeldada seda, kuidas alamülesanded on üksteisega seotud ning õppida vaid neid mooduleid, mida veel pole õpitud või inimene kirjeldada ei oska. Modulaarsetest närvivõrgu arhitektuuridest on varem teiste hulgas kirjutanud Coline Devin stiimulõppega roboteid kontrollima õppides ja Jacob Andreas visuaalsetele küsimustele vastamisega.

1.3.1 Robotite kontrollimine

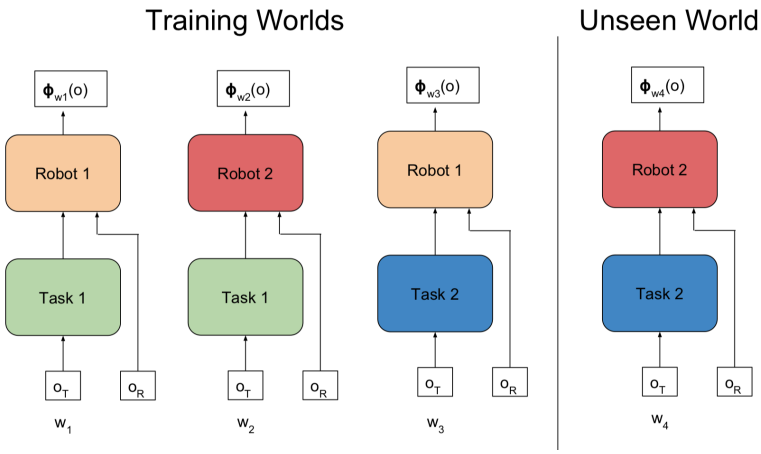
Devin uuris artiklis "Learning Modular Neural Network Policies for Multi-Task and Multi-Robot Transfer"[6] erinevate ülesannete või roboti spetsiifiliste moodulite kombineerimist. Ülesanne oli õppida füüsikalises simulatsioonis stiimulõppega tõmbama sahtlit või lükkama kuubikut ning robotid olid erinevate kujude ja vabadusastmetega käed. Kui ülesande moodul on piisavalt agnostiline roboti suhtes, siis õpitakse robotitele edastama ainult ülesande spetsiifilist infot. Vastupidiselt, kui roboti moodul on agnostiline ülesande suhtes, siis õpitakse kontrollima robotit ennast. Nii suudetakse näiteks nelja liigesega robotikäega lükata kuubikut, kuigi treenimise käigus sellist maailma kunagi ei eksisteerinud (joonis 2).

Ülesandespetsiifiline moodul koosnes konvolutsioonilistest ja *spatial softmax* närvivõrgu kihtidest, mille ülesanne oli anda robotile inimese poolt täpsustamata kujul

informatsiooni ülesande sisu ja väljaku kohta. Robotispetsiifiline moodul pidi oskama täielikult ühendatud kihtide abil seda infot tõlgendada ning ennast hästi kontrollima, et ülesannet õigesti lahendada. Mooduleid kombineeriti nagu näidatud joonisel 3).



Joonis 2. Robotid ja ülesanded [6].



Joonis 3. Moodulite kombineerimine [6].

Selleks oli moodulite õppimiseks vaja eraldada kaofunktsioon robotist sõltuvast sisemiseks ja ülesandest sõltuvast välimiseks kaofunktsiooniks. Samuti täheldati, et

selline õppimine peab toimuma kõikide olemasolevate maailmadega paralleelselt - ei ole võimalik õppida ülesandeid robotitega lahendama ükshaaval. Kui ülesandeid või roboteid on liiga vähe, siis on oht ülesobitamiseks. Robotimoodul õpitakse hästi ära, kui ülesandeid on palju, ja ülesandemoodul paljude robotitega.

1.3.2 Visuaalsete küsimuste vastamine

Ronghang Hu, Trevor Darrell ja Jacob Andreas tegelesid artiklis "Explainable Neural Computation via Stack Neural Module Networks"[11] modulaarsete närvivõrkudega, mille vahepealseid väljundeid on lihtne inimesel tõlgendada. Põhiline rõhk oli visuaalsete küsimuste vastamisel (VQA) CLEVR [12] andmetel. Loodi moodulid, mis saavad sisendiks tähelepanu all olevad objektid ning rakendavad kindlat operatsiooni ilma sisendita või siis argumendiga. Näiteks *find* moodul võib saada argumendiks värvi ning peab eemaldama kõik tähelepanu all olevad teist värvi objektid.

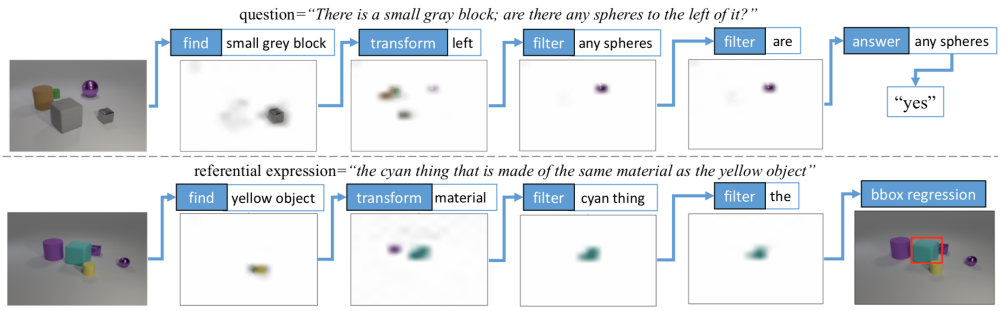
Kogu mudel koosnes neljast peamisest komponendist:

1. Tekstikodeerija, mis saab inimese poolt esitatud küsimuse sisendiks ning kodeerib selle mudelile arusaadavaks. Kasutati kahesuunalist LSTM mudelit (BiLSTM).

2. Moodulite arhitektuuri kontroller, mis vastavalt küsimusele (1. komponendi väljund) ja ajahetkele otsustab, millist treenitud moodulit tõenäoliselt võiks kasutada. Erinevad ajahetked tegelevad erinevate küsimuse osade tõlgendamisega. Teise väljundina luuakse tekstiline parameeter, kus talletatkse vajaduse korral lisainformatsiooni moodulile.

3. Konvolutsiooniline võrk, mis tõlgendab moodulitele sisendpilti.

4. Usalduse skoori 2. komponendist ja moodulite väljundite korrutiste keskmine tagastatakse ajahetke väljundina. Viimase ajahetke väljund vastab küsitud küsimusele. Kusjuures on iga ajahetke väljundit võimalik visualiseerida (joonis 4).

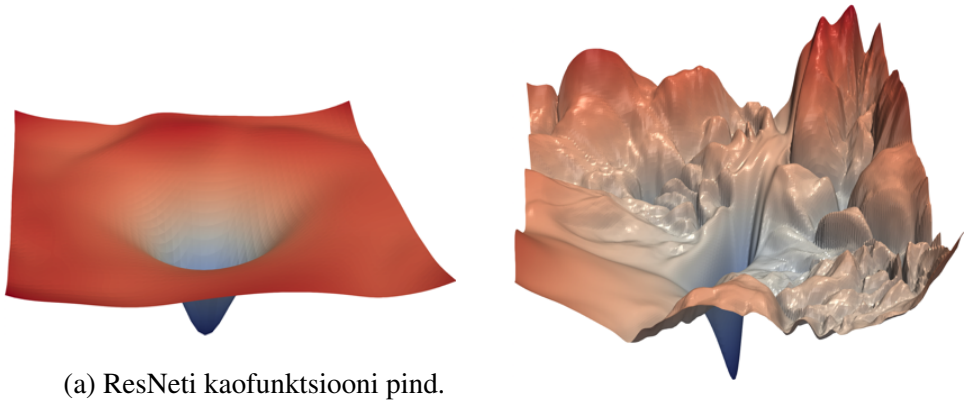


Joonis 4. Vahepealsete väljundite visualiseerimine töös [11].

Töös kasutati mäluga närvivõrke [20]. Samuti selgitati vajadust mooduleid eraldi eeltreenida. Kui prooviti sellise arhitektuuriga alustada kõikide moodulite treenimist nullist, siis õppimist kas ei toimunud või oli aeglane. Autorite sõnul oli mudelil keeruline õppida üheaegselt kaale parandama ja võimalikku arhitektuuri ennustama.

1.4 Alamülesannete õppimine

2015. aastal avaldatud tehisnärvivõrgu arhitektuur ResNet [9] oli unikaalne selle poolest, et see oli palju sügavam kui varem loodud närvivõrgud. ResNeti osad arhitektuurid kasutasid üle saja kihi oma mudelis. Ometi suutis see üldistada probleemi lahendamist hästi, sest seal kasutati vahelejätmisühendusi (*skip connections*). Autorite sõnul suutis ResNet nii õppida alamülesandeid. On näidatud, et sellest tulenevalt on kaofunktsiooni pind palju sujuvam (joonised 5a, 5b).



(a) ResNeti kaofunktsiooni pind.

(b) ResNeti kaofunktsiooni pind ilma *skip* ühendusteta.

Joonis 5. ResNeti kaofunktsiooni pinnad [17].

Mittesujuval kaofunktsioonil on mitmeid halbu omadusi, kuid kõrgdimensionaalsete andmete puhul lokaalsetesse miinimumidesse jõudmine ei ole üks neist [14]. Sagedasem oht on sadulpunktidesse jõudmine, mispärast võtab õppimine palju aega. Sügavate närvivõrkude puhul, kus kaofunktsioon ei ole sujuv, on oht sattuda nn "halba sadulpunkti". Halvad sadulapunktid on punktid, kus Hesse maatriksi kõik liikmed on mittenegatiivsed ja kõik esimese astme tuletised on 0. On ilmne, et sellistel punktides ei saa õppimist traditsiooniliste algoritmidega toimuda.

Alternatiivselt võib aga veenduda, et halbasid sadulpunkte ei oleks üldse. On tõestatud, et kolmekihilisel närvivõrgul - st närvivõrgul ühe peitkihiga - selliseid punkte ei ole [14]. Teoreetiliselt on võimalik neurovõrke treenida ülevalt alla ühe kihi haaval, kus iga treenitava kihi jaoks on oma treeningandmestik ning treenitud kihi kaale enam muuta ei saa. Nii hoitakse kaofunktsioon sujuv ning valmiv mudel ülesobitaks vähe, sest luuakse üldistatud lineaarne regressioon.

Siinkohal on kasulik märkida, et eelnevalt kirjeldatud viisi närvivõrkude treenimiseks ei saa praktikas kasutada. Närvivõrgud on erilised selle poolest, et need oskavad leida seoseid, mida neile sisse ei kodeerita. Kui närvivõrku treenida kiht-kihi haaval, siis see

tähendaks, et on tarvis täpselt teada, mis on optimaalne treeningandmestik ning millega iga kiht peab tegelema. Selline tegevus on sügavate närvivõrkude puhul keeruline ning kaotab masinõppimise mõtte.

Kui aga eeldada, et probleemides sageli peitub hierahilisus, siis annab see õigustuse õppida alamülesannete kaudu. Sellisel juhul on optimaalne närvivõrk leitav nii tervikliikuna kui ka modulaarselt õppides. Modulaarse õppimisega, kus treenitavad moodulid on madalakihilised, eeldatakse, et oht (halbadeks) sadulpunktideks on väiksem, sest et oht leida mürast tulenevaid ebaolulisi seoseid on väiksem. Peatükis 3.3 näidatakse empiiriliselt, kas modulaarse närvivõrgu kaofunktsioon on ka sujuvam.

2 Ülesande ja lahenduse kirjeldused

2.1 Ülesande valik

Ülesanne, mis kõige paremini kirjeldaks modulaarsust, tekkis töö käigus. Oluline oli, et õpitav ülesanne oleks lihtsasti tõlgendatav ja visualiseeritav. Selleks sai palli liikumise ennustamine kaadrite kaupa lihtsustatud simulatsioonides, mida täpsemalt kirjeldatakse järgmises peatükis.

Alternatiivselt võinuks kasutada sarnaselt peatükis 1.3 kirjeldatud töödele füüsikaliseks modelleerimiseks laialt rakendust leidvaid füüsika- või graafikamootoreid, kuid et kiiremini jõuda töös oluliste ideede kajastamiseni, oli kasulik luua alustuseks veel lihtsamaid mooduleid ilma pildi tõlgendamiseta.

Praeguse ülesande valiku parim omadus on selle lihtne skaleerimine: töös õpitud mooduleid saab taaskasutada ka uutes füüsikamootoreid kasutatavates simulatsioonides, kuhu on lisatud hõõrdejõud ja pallide pöörlemine. Nii on lihtne tööd edasi arendada ja muuta õpitavat ülesannet üha keerulisemaks.

2.2 Probleemi püstitus

Ülesanne on ennustada diskreetse ajaga piljardipalli liikumist ning ideaalseid põrkeid vastu seinu tasapinnal, kus puuduvad hõõrdejõud. Sisendiks on mudelil pallide x ja y koordinaadid koos kiirusega suvaliste seinte koordinaatidega vahemikus $b_1 \in [-10,-1], b_2 \in [1,10]$ tasapinnal. Sellised seinad genereeritakse x ja y-teljel juhuslikult. Pall saab vabalikult takistuseta liikuda seinte vahel. Eesmärgiks on mudel, mis suudaks ennustada piljardipalli liikumist mitu kaadrit ette.

Kui inimene õpib uusi probleeme lahendama, kasutab ta selleks varem õpitud arutlemist või alamülesandeid [1]. Sarnaselt eeldame, et treenitav mudel, mis ennustamisega tegeleb, pääseb ligi erinevatele varem treenitud moodulitele, mis võivad antud ülesandega

aidata.

Et luua olukord, kus sellised moodulid saaksid eksisteerida ilma klastritesse kogumisteta või arhitektuuriotsingu algoritmideta, treenitakse neid mooduleid eraldi koos selleks loodud treeningandmete ja -märgenditega. Luuakse andmestikud sirgliikumise ja erinevate põrgete jaoks.

2.3 Treeningandmestiku loomine

Ühedimensionaalne treeningandmestik luuakse mudelite treenimise ajal generaatoridega, mis loovad mudelile treenimisaegselt miniplokke. Generaatoreid kirjeldab tabel 1.

Tabel 1. Treeningandmete generaatorid. (*) Kõik jaotused olid ühtlased, väljaarvatud $b \in \{b_1, b_2\}$ valitakse juhuslikult.

tunnus	minimaalne väärtus	maksimaalne väärtus
alumine äär b_1	-10	-1
ülemine äär b_2	1	10
põrkele lähem äär b	b_1	b_2
kiirus v	0	$0.1 \cdot sign(b)$
algkoordinaat x	b	$b - 2v$

Kui generaatori kaks väljundit ühendada, siis on tegemist kahedimensionaalse pinnaga. Funktsioon f^*, mis lahendab sellise ülesande ideaalselt on:

$$f^*(x,v,b_1,b_2) = \begin{cases} x+v, \text{ehk } \mu(x,v), & \text{kui } |x+v| < |\beta(x,b_1,b_2)| \\ 2\beta(x,b_1,b_2) - x - v, \text{ehk } \rho(x,v,\beta(x,b_1,b_2)), & \text{kui } |x+v| \geq |\beta(x,b_1,b_2)| \end{cases},$$

kus

$$\beta(x,b_1,b_2) = \begin{cases} b_1, & \text{kui } \frac{b_2-x}{b_2-b_1} > 0.5 \\ b_2, & \text{kui } \frac{b_2-x}{b_2-b_1} \leq 0.5 \end{cases},$$

Funktsioon μ tegeleb sirgliikumise kujutamisega, ρ põrgetega, β lähima ääre valikuga ja juht $|x+v| \geq |\beta(x, b_1, b_2)|$ käitub tähelepanuna, mis näidatab, millist alamülesannet on tarvis lahendada. Juhte, kus $sign(b_1) = sign(b_2)$ selle töö raames ei vaadelda.

2.4 Eeldused

Tasub uurida, milliste eelduste tegemine aitab mudelil kiiremini õppida. Kas

- mooduli kaupa õppimine on kiirem kui kõiki mooduleid samaaegselt õppida,
- tähelepanu moodulid on möödapääsmatud ülesande lahendamisel ning
- kas tähelepanu (peatükk 2.5.1) peab olema *hard attention* (edaspidi absoluuttähelepanu).

Võrdlemaks üleval kirjeldatud ideid, treenitakse eraldi

1. *end-to-end* närvivõrku, mis saab sisendiks algsed treeningandmed ning millest saab lähtemudel, millega võrrelda teisi mudeleid;
2. mooduleid kasutav närvivõrku, mis kasutab varem treenitud mooduleid, kuid ei kasuta mehhanisme, millega tagasilevi algoritmi ei saaks enam teha (sh kõva tähelepanu);
3. närvivõrku, mis kasutab mooduleid ning aitavaid eeldusi, mille arhitektuur on kindlaks määratud ning kus õppimist ei saa triviaalselt toimuda.

Kui teine mudel on treenitud piisavalt hästi, siis võiks see saada võrdseid tulemusi kolmandaga. Samuti võiks olla see kergemini rakendatav järgmistes ülesannetes (nt. olla ise moodul, mida saab õppida).

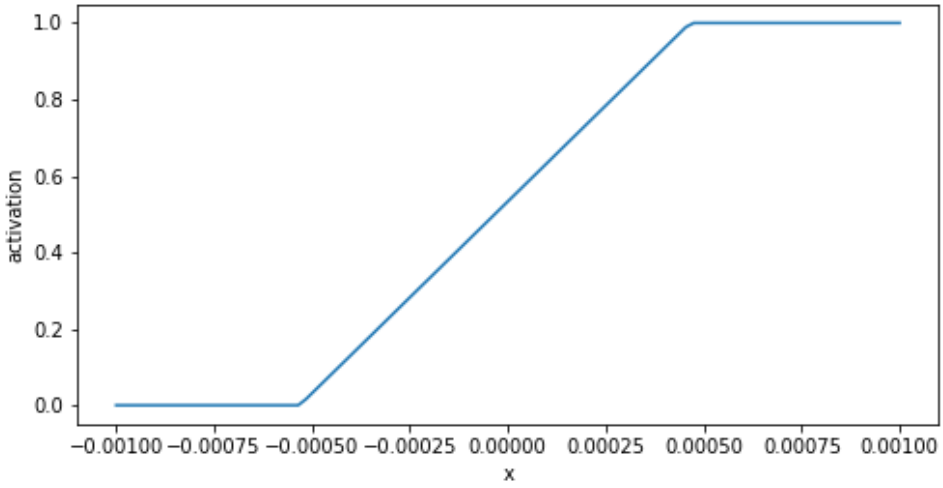
2.5 Treenitud moodulid

2.5.1 Tähelepanu mehhanism

Tähelepanu eesmärk on mudelis tunda ära, kas on vaja ennustada sirget liikumist või põrget. Sarnast tähelepanu kasutatakse teises kontekstis, kus on ülesandeks lähema seina ära tundmine mõlema telje puhul. Tähelepanu käitub lülitina, mis laseb ühe kahest ennustusest läbi.

Et oleks abi alamülesandeid lahendavatest moodulitest, siis on tarvilik omada mingit tähelepanu alamülesande tuvastamiseks. Tähelepanu moodul peab seejuures olema diferentseeritav, et kogu mudelile saaks hiljem tagasilevi algoritmi rakendada. Sellest tulenevalt oli kasulik seda realiseerida nii, et tähelepanu oleks võimalikult lähedal väärtustele 0 ja 1 nagu on seda ka absoluuttähelepanu (joonis 6). absoluuttähelepanui puhul ei toimuks õppimist, vaid oleks ette määratud funktsioon, mis sisuliselt tegeleb ülesande klassifitseerimisega.

Eeldades, et tähelepanu toimib õigesti, ei ole tarvis teada, kuidas moodulid toimivad juhtudel, kus neid ei rakendata. Nii saavad moodulid olla vähemate treenitavate parameetritega ning seetõttu ka lihtsamad.



Joonis 6. Tähelepanu aktivatsioonifunktsioon.

Tähelepanu implementatsioon nägi välja kahe järjestikuse neuronina, millel olid vastavalt ReLU ja sigmoidi aktivatsioonifunktsioonid. Tähelepanu käitub mudelis justkui lüliti, mis vastavalt sisendile laseb läbi kas esimese sisendi või teise sisendi. Ideaalis on tähelepanu a järgneval kujul:

$$a(x) = \begin{cases} 1, & \text{kui } x \geq b \\ 0, & \text{kui } x < b \end{cases},$$

kus b on esimese neuroni parameeter. Keerulisemate tähelepanude juures oleks b arvutamiseks vaja rohkemat kui üht neuronit. Tähelepanu sisendi x ja kahendi sisenditensori y, z puhul käitub tähelepanu funktsioonina

$$f_a(x,y,z) = a(x)y + (1 - (a(x))z. \tag{2.1}$$

2.5.2 Ääre valik, põrge ja sirgliikumine

Neli selgelt eristatavat alamülesannet, mida õppida vajadusel tähelepanu abiga lahendama ühes dimensioonis on:

1. Ääre valik β: vastavalt palli x ja kahe ääre koordinaadile b_1, b_2 valida tähelepanu moodulit kasutades äär b, kus $b = b_1$ juhul, kui $\frac{b_2 - x}{b_2 - b_1} > 0.5$ ning vastasel juhul $b = b_2$;

2. Põrkega liikumine ρ: ennustada järgmise kaadri koordinaat, saades sisendiks palli koordinaadi x, kiiruse v ja lähima seina koordinaadi b. Oodatav väljund on $2b - v - x$. Õpitavad parameetrid on u_1, u_2, u_3 funktsioonis $o_\rho = u_1 x + u_2 v + u_3 b$.

3. Sirglikumine μ: ennustada järgmise kaadri koordinaat, saades sisendiks x ja kiiruse v. Oodatav väljud on $x + v$. Närvivõrgu puhul on mooduli õpitavad parameetrid w_1, w_2 funktsioonis $o_\mu = w_1 x + w_2 v$.

4. Lüliti σ põrke või sirgliikumise mooduli väljundi valimiseks: tähelepanu mooduli abil ennustada, kas lähima seinaga b, palli praeguse koordinaadiga x ja kiirusega v on järgmine kaader pärast põrget või mitte. Oodatav tähelepanu a^* väljund on

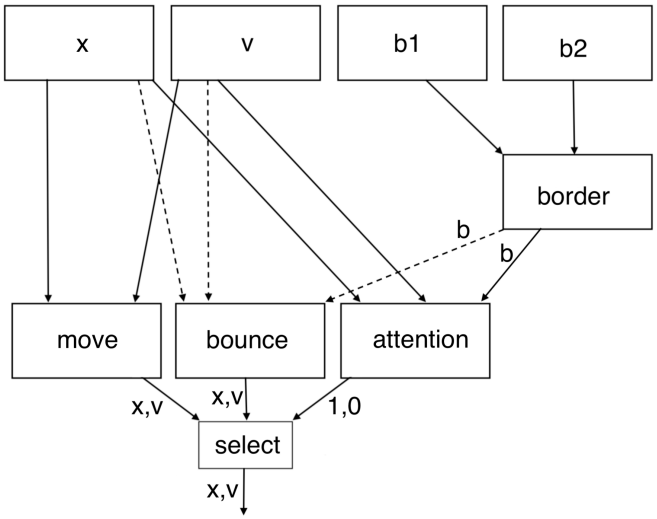
$$a^*(x, v, b) = \begin{cases} 1, & \text{kui } |x + v| - |b| \geq 0 \\ 0, & \text{kui } |x + v| - |b| < 0 \end{cases}$$

ning $\sigma(x, v, b) = a^*(x, v, b) o_\mu(x, v, b) + (1 - a^*(x, v, b)) o_\rho(x, v, b)$, kus $b = \beta(x, b_1, b_2)$.

2.6 Treenitud mudelid

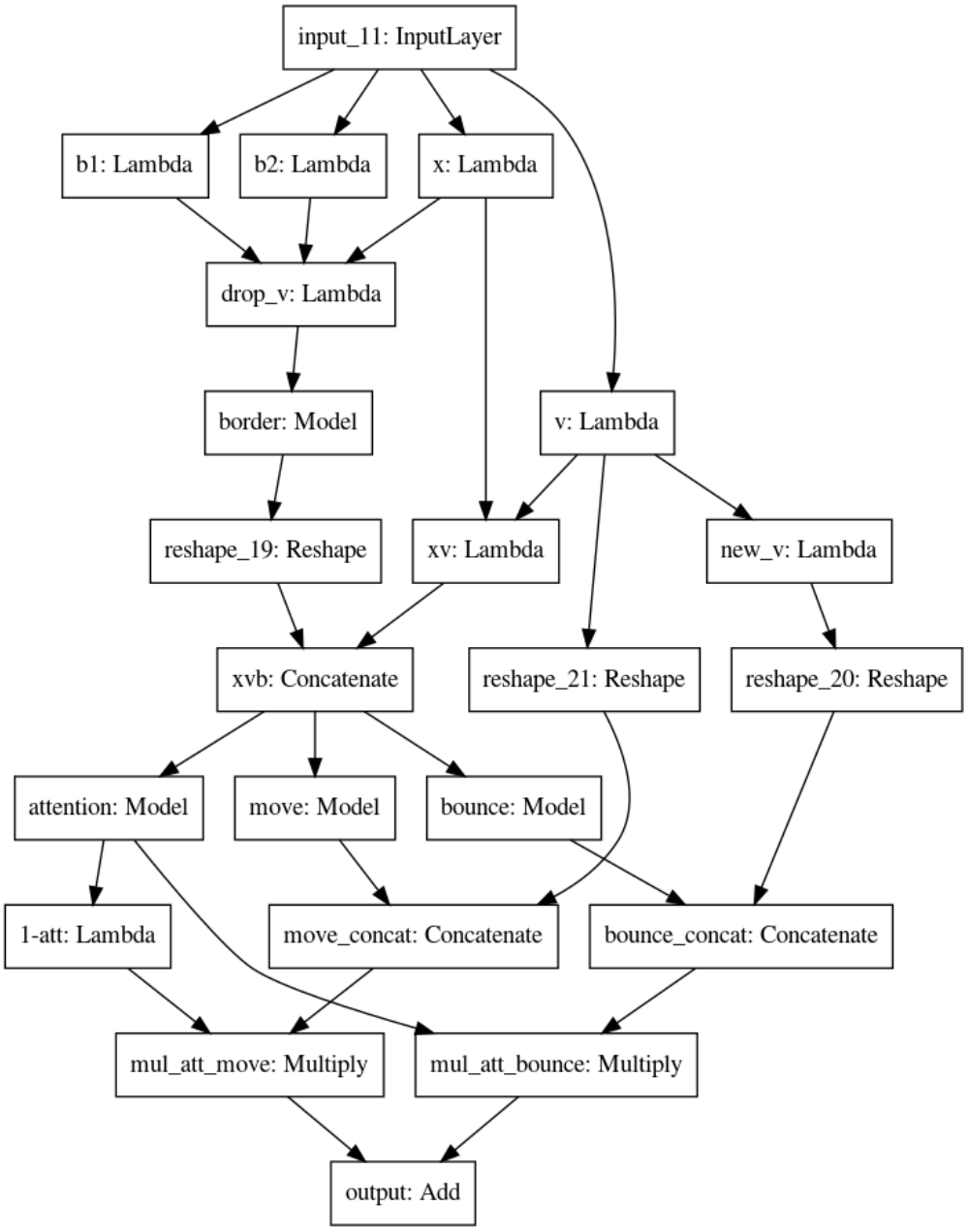
2.6.1 Modulaarne arhitektuur

Peamooduli arhitektuur loodi eelnevaid mooduleid kombineerides, et ennustada ühes dimensioonis palli liikumist ja põrkamist. Kui kasutada eeltreenitud mooduleid, siis pole mudelil eraldi õppimist tarvis sarnaselt *zero-shot* õppimisele. Eelmises peatükis kirjeldatud moodulite kombineerimine käis sarnaselt joonisele 7.



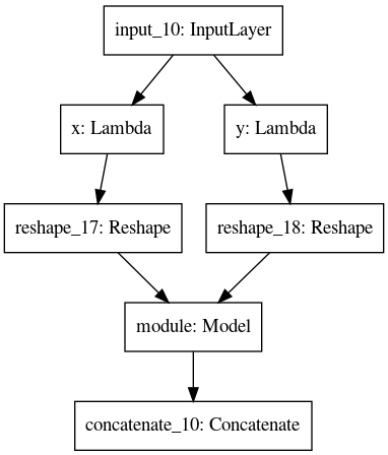
Joonis 7. Moodulite kombineerimine lihtsa diagrammiga, kus on *border* β, *move* μ, *bounce* ρ ja *select* σ.

Valemilt 2.1 võib mõista, miks tähelepanu peab olema sarnane absoluuttähelepanu loogikale - vastasel juhul, kui tähelepanu väljund ei ole 1 ega 0, siis hiljem põrkemooduli väljund ja sirgliikumise mooduli väljundeid kokku liites on tulemus ebatäpne. Seepärast on kasulik jätta tähelepanu sellisel kujul treenituks isegi juhul, kus teiste moodulite kaale algväärtustatakse uuesti. Arhitektuuri kujutus Kerases [4] on joonisel 8.



Joonis 8. Peamooduli arhitektuur.

Et ennustada palli liikumist kahedimensionaalsel pinnal, on piisav mudelil sisend poolitada ning kasutada kahel korral peamoodulit ning väljundid jälle ühendada nagu näidatud joonisel 9.



Joonis 9. Kahedimensionaalse liikumise mudel, kus Lambda kihid on vastavad koordinaadid koos kiirustega ning module tähistab peamoodulit.

Sellise arhitektuuriga närvivõrk treeniti kahel eri viisil. Esimesel juhul kasutati eeltreenitud mooduleid ning mudelit ennast ei treenitud. Teisel juhul algväärtustati iga mooduli väljaarvatud tähelepanu kaalud uuesti ning õpiti kogu mudel algusest. Kõiki mudeleid ja mooduleid treeniti Adam optimiseerijaga Kerase vaikeparameetritega [15].

2.6.2 *End-to-end* närvivõrk

End-to-end närvivõrk loodi ja treeniti sarnaselt baasmudelile, millega võrreldi teisi mudeleid füüsikaliste kehade liikumise kujutamiseks töös "Interaction Networks for Learning about Objects, Relations and Physics"[2]. Et töös kasutatav simulatsioon on lihtsam, kasutatakse kahe 200 neuronilise kihi asemel kahte 128 neuronilist kihti.

Esimeses kihis kasutati ReLU aktivatsiooni, et oleks võimalik eraldada kahte alamülesannet. Nii õpiks selline võrk ideaalis jaotama oma neuroneid kahe alamülesande jaoks eraldi, ühel juhul on ühed aktivatsioonid 0, teisel juhul teised.

Teises kihis kasutati lineaarset aktivatsiooni, et oleks võimalik teha ennustusi ka negatiivses piirkonnas. Eeldatavasti võtab sellise närvivõrgu treenimine oluliselt kauem aega.

3 Tulemused

3.1 Moodulite kaotused

Iga mooduli ja mudeli puhul (v.a tähelepanu ja äärevalik) oli kaofunktsiooniks keskmine ruuthälve (MSE), kus m treeningolemi kadu on $\Lambda(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{m}\sum_i (y_i - \hat{y}_i)^2$. Treenitud moodulid koos treenimisaegade ja vigadega on toodud tabelis 2.

Tabel 2. Moodulite kaotused. Andmaks treenimise ajale konteksti, treeniti mooduleid ja mudeleid Kerases Nvidia graafikakaardil GTX 1070. (*) Et tähelepanu treeniti standardse normaaljaotusega generaatoril, siis tuhande testolemiga ei olnud ükski vahemikus $(-0.0005, 0.0005)$, kus olnuks viga nagu näha jooniselt 6. (**) Ääre valiku moodulit treeniti binaarse rist-entroopiaga kaofunktsioonil, kus oli saavutatud täpsus 99.1%.

moodul	treenimise aeg (min)	parameetrite arv	MSE
kiirus v	4	35	1.14e-6
tähelepanu a	5	4	0*
äärevalik β	5	3	**
sirgliikumine μ	10	3	1.22e-7
põrge ρ	10	3	1.20e-7

3.2 Mudelite kaotused

Treenitud mudeleid on kolm:

1. *end-to-end* närvivõrk (peatükk 2.6.2), mis on baasmudel, millega võrrelda teisi närvivõrke.

2. modulaarne närvivõrk (peatükk 2.6.1), mis on eeltreenitud moodulitest kombineeritud ning mis ei vaja eraldi lisaks õppimist.

3. samasugune modulaarne närvivõrk, mille moodulite kaalud väljaarvatud tähelepanu on algväärtustatud uuesti, et õppida tervet ülesannet korraga lahendama.

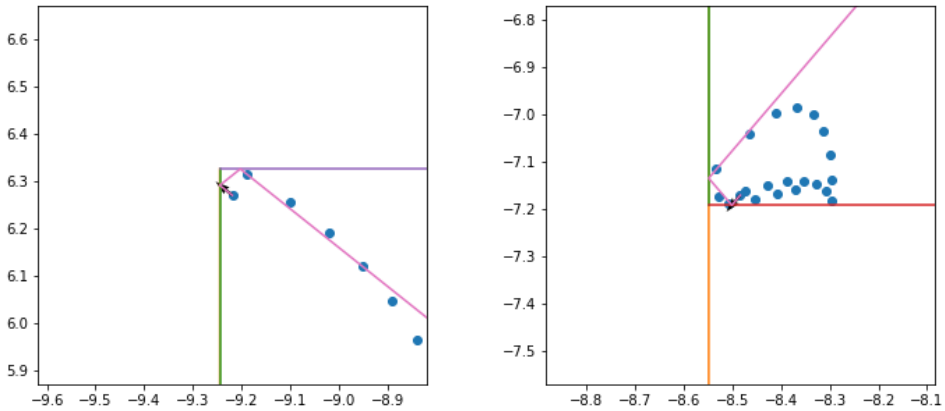
Peatükis 2.2 püstitatud hüpotees, et absoluuttähelepanu võib olla tarvilik edukaks õppimiseks, lükati ümber, sest tähelepanu *a* sai piisavalt häid tulemusi. Küll aga on põhjust olla kriitiline kaofunktsiooni valiku osas, sest üldine ruuthälbe kaofunktsioon ei väljenda piisavalt hästi ülesannete lahendamise oskust. *End-to-end* mudel saavutas sarnase vea algväärtustatud modulaarsele närvivõrgule, kuid ennustas liikumist äärte lähedal ebatäpselt. Efektiivsem olnuks kirjutada eraldi kaofunktsioon, mis määrab piiridest väljunud ennustusele suurema vea.

Kolme mudeli tulemused on toodud tabelis 3.

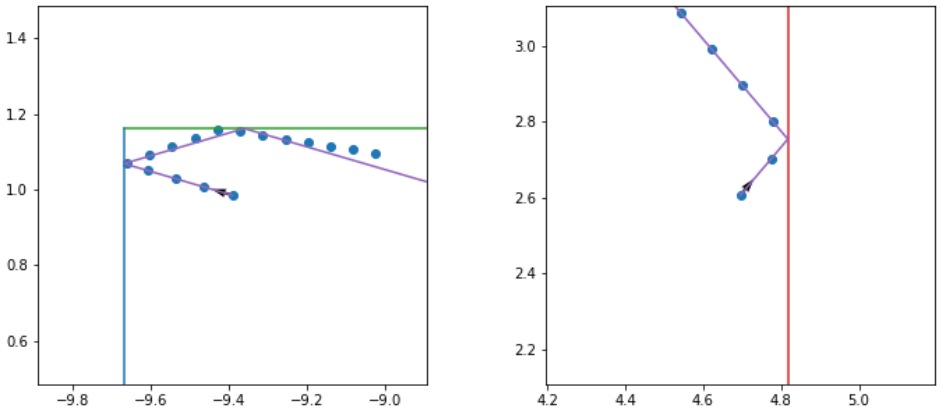
Tabel 3. Mudelite kaotused. Andmaks treenimise ajale konteksti, treeniti mooduleid ja mudeleid Kerases Nvidia graafikakaardil GTX 1070. (*) Siin on loetud treenimise ajaks moodulite treenimist.

mudel	treenimise aeg (min)	parameetrite arv	MSE
end-to-end	85	18 196	1.03e-4
eeltreenitud modulaarne	34*	73	1.85e-4
algväärtustatud modulaarne	22	73	2.05e-4

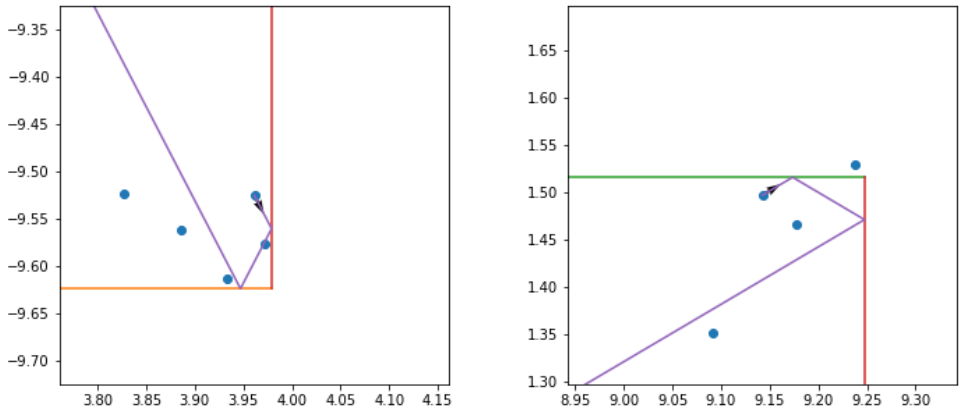
Visualiseerimaks õpitud mudeleid, kujutatakse palli liikumist äärte lähedal. *End-to-end* mudeli, eeltreenitud mudeli ja modulaarse arhitektuuriga mudeli ennustused on vastavalt joonistel 10, 11, 12.



Joonis 10. *End-to-end* mudeli ennustused. Eksimist on sagedamalt.



Joonis 11. Eeltreenitud moodulitega mudeli ennustused.

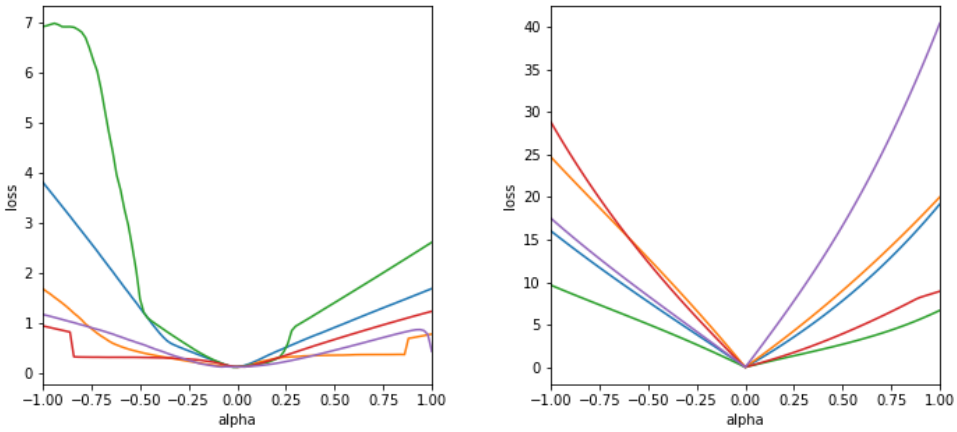


Joonis 12. Algväärtustatud modulaarse mudeli ennustused.

Põhjus, miks algväärtustatud kaaludega modulaarne mudel õpib kehvasti, võib tuleneda võrdlemisi kehvast kaofunktsiooni tasandist. Rohkemate neuronitega mudeleid loetakse tihti liiasusega mudeliteks (*redundant*), sest osad neuronid õpivad samu ülesandeid lahendama. See ei ole lõppmudeli jaoks optimaalne, kuid aitab õppimisele kaasa [18].

3.3 Mudelite ühedimensionaalsed kaotustasandid

Kaotustasandeid kujutatakse ühedimensionaalselt sarnaselt tööle [17]. Selleks muudetakse treenitud parameetrid θ^* ning igat kaalu muudetakse mingi suvalise arvuga, näiteks normaaljaotusest võetud arvuga, kus keskmine $\mu = 1$ ja standardhälve $\sigma = 0.5$. Pärast muutmist saadakse uued kaalud θ' ning kaotust $\Lambda(f(\mathbf{x}, \theta(\alpha)), \hat{y})$ kujutatakse argumendiga vahemikus $\alpha \in [0,2]$, kus $\theta(\alpha) = (1-\alpha)\theta^* + \alpha\theta'$. Kaotustasandit võib kujutada kui kaotuse ühedimensionaalset ristlõiget ruumis $\mathbb{R}^n$, kus n on õpitavate parameetrite arv.



(a) Kaotuse pind modulaarses närvivõrgus. (b) Kaotuse pind *end-to-end* närvivõrgus.

Joonis 13. Kaotuse pinnad.

Kaotuspinnad joonistel 13a, 13b vihjavad sellele, et modulaarses närvivõrgus algväärtustatud kaaludega ei toimu õppimist kiiremini. Küll aga on need mudelid põhjapaneva väite tegemiseks liiga erinevad. Eelnevalt mainitud idee, et baasmudel kasutab liiaseid neuroneid, võib mõjutada seda, et kaofunktsioon paistab sujuvam. ResNeti puhul (joonised 5a, 5b) oli kahe mudeli arhitektuur täpselt sama ja kaotuspindu oli lihtsam võrrelda.

Kokkuvõte

Töös uuriti modulaarsete närvivõrkude efektiivsust ülesannete lahendamisel ning näidati, kuidas närvivõrkude mudelitele inimteadmisi või eeldusi juurde lisada. Selleks koostati närvivõrk, mis koosnes eeltreenitud moodulitest, millest igaüks oskas lahendada teatud alamülesannet, ning võrdlemiseks treeniti tervet ülesannet korraga õppiv rohkemate parameetritega, kuid ilma eeldusteta mudel.

Modulaarsete mudelite arendamise käigus selgus, et vajalik on implementeerida tähelepanu, millega eristada erinevaid alamülesannete juhte. Sama tähelepanu implementatsioon oli korduvalt kasutuses erinevates alamülesannetes. Tähelepanul on suur roll, et modulaarselt õppida keerukamaid ülesandeid lahendama.

Kui baasmudel ei suutnud eristada alamülesandeid ja minimeeris tehtavat viga ennustades midagi kahe ülesande vahepealset, siis tähelepanuga moodulid said hästi hakkama erinevate alamülesannetega. Nõrgaks kohaks jäi siinkohal liiga üldise kaofunktsiooni valik, kus vea suurus ei kujutanud eriti efektiivselt mudelite oskust lahendada ülesandeid: *end-to-end* võrk sai pärast poolteist tundi treenimist keskmiseks ruuthälbeks $1.03 \cdot 10^{-4}$ ühikut füüsikalisel tasandil ning eeltreenitud moodulitega võrk $1.85 \cdot 10^{-4}$ ühikut.

Samuti visualiseeriti treenitud mudelite kaotuspindu, et leida teoreetiline õigustus modulaarsete võrkude kasutuseks. Püstitatud hüpoteesi, et modulaarse arhitektuuri kaotuspind on sujuvam, ei näidatud, sest treenitud närvivõrkude arhitektuurid erinesid liialt palju. Seega on vara öelda, et see oleks põhjus hakata närvivõrkude arhitektuure muutma teistes ülesannetes.

Töös pakutud ideid võiks edasi arendada püüdes õppida keerulisemaid ülesandeid lahendama modulaarselt, kus mudelil tuleb valida tähelepanu abil, milliseid mooduleid valida, et edukamalt õppida. Sellised keerulisemad ülesanded võiksid olla valdkonnas, kus on kindlasti alamülesanded, mille omavahelisi seoseid oskab inimene mudelile ette anda.

Viidatud kirjandus

[1] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv e-prints*, page arXiv:1806.01261, Jun 2018.

[2] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu. Interaction Networks for Learning about Objects, Relations and Physics. *arXiv e-prints*, page arXiv:1612.00222, Dec 2016.

[3] T. Ben-Nun and T. Hoefler. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. *arXiv e-prints*, page arXiv:1802.09941, Feb 2018.

[4] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015. Viimati väisatud 10.05.2019.

[5] L. Cruz, S. V. Buldyrev, S. Peng, D. L. Roe, B. Urbanc, H.Štanley, and D. L. Rosene. A statistically based density map method for identification and quantification of regional differences in microcolumnarity in the monkey brain. *Journal of Neuroscience Methods*, 141(2):321 – 332, 2005.

[6] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine. Learning Modular Neural Network Policies for Multi-Task and Multi-Robot Transfer. *arXiv e-prints*, page arXiv:1609.07088, Sep 2016.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] S. J. Hanson and L. Y. Pratt. Comparing biases for minimal network construction with back-propagation. In D.Š. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 177–185. Morgan-Kaufmann, 1989.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, Dec 2015.

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints*, page arXiv:1207.0580, Jul 2012.

[11] R. Hu, J. Andreas, T. Darrell, and K. Saenko. Explainable Neural Computation via Stack Neural Module Networks. *arXiv e-prints*, page arXiv:1807.08556, Jul 2018.

[12] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *arXiv e-prints*, page arXiv:1612.06890, Dec 2016.

[13] A. Karpathy. Convolutional neural networks for visual recognition. `http://cs231n.github.io/convolutional-networks/`, 2015. Viimati väisatud 10.05.2019.

[14] K. Kawaguchi. Deep Learning without Poor Local Minima. *arXiv e-prints*, page arXiv:1605.07110, May 2016.

[15] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec 2014.

[16] R. Kurzweil. *How to Create a Mind: The Secret of Human Thought Revealed*. Penguin Books, New York, NY, USA, 2013.

[17] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the Loss Landscape of Neural Nets. *arXiv e-prints*, page arXiv:1712.09913, Dec 2017.

[18] T. Matiisen. What neural networks actually do? https://neuro.cs.ut.ee/what-neural-networks-actually-do/, Aug 2018. Viimati väisatud 10.05.2019.

[19] K. Rupp. CPU, GPU and MIC Hardware Characteristics over Time. https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/, 2016. Viimati väisatud 10.05.2019.

[20] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-To-End Memory Networks. *arXiv e-prints*, page arXiv:1503.08895, Mar 2015.

[21] Y. Tao. State-of-the-art result for all machine learning problems. https://github.com/RedditSota/state-of-the-art-result-for-\machine-learning-problems, 2017. Viimati väisatud 10.05.2019.

[22] Wikimedia Commons. Perceptron unit. https://commons.wikimedia.org/wiki/File:Perceptron-unit.svg, 2014. Viimati väisatud 10.05.2019.

Lisad

I. Koodi repositoorium

Lingilt https://github.com/Muruniiduk/modularnns leiab juhise, kuidas töös saadud tulemusi reprodutseerida.

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Jaan Erik Pihel**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

   Modulaarsed närvivõrgud,

   mille juhendajad on Meelis Kull ja Markus Kängsepp,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, alates 10.05.2019 kuni autoriõiguse kehtivuse lõppemiseni.

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 10.05.2019