TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ralf Anari IASB142875

SÜVAÕPPE ALGORITMIDE VÕRLEMINE KASUTADES TENSORFLOW KERASE TEEKI

Bakalaureusetöö

Juhendaja: Eduard Petlenkov
Phd

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Ralf Anari

20.05.2019

Annotatsioon

Käesoleva bakalaureuse töö eesmärgiks on uurida ning võrrelda populaarseid süvaõppe algoritme ja arhitektuure. Nende võrdlemiseks genereeriti ja treeniti erinevate sügavuste ning kihi elementide arvuga närvivõrgud benchmark andmebaasidel ning genereeriti treeningu ajalugudest heatmapid. Edasine uurimine toimus heatmappidelt loetava informatsiooni baasil. Kõik benchmark andmebaasid olid võetud Keras teegist ning kõik arhitektuurid olid samuti implementeeritud kasutades Keras teeki Jupyter notebookis. Treening keskkonnaks olid Google colaboratory ning personaalarvuti. Andmebaasideks valiti pildi klassifikatsiooni andmebaasid ning rekurrentsete närvivõrkude võrdlemiseks IMDB ja Reutersi andmebaas. Töö tulemuseks illustreeriti pärilevi närvivõrkude nõrkused pildi klassifitseerimisel, konvolutsiooniliste närvivõrkude kalduvus ülesobitamisele ning selle probleemiga lahenamiseks kasutatavaid populaarseid meetodeid. Samuti illustreeriti GRU ja LSTM võimekus sentimenti analüüsil ning teksti klassifitseerimisel väga väikeste mõõtmetega arhitektuurides ning nende erinevused võimekuses andmete täidiste asukohtade muutuses.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 23 leheküljel, 6 peatükki, 41 joonist, 0 tabelit.

Abstract

Comparison of Deep Learning Algorithms Using Tensorflow Keras library

The purpose of his bachelors thesis is to investigate and compare popular deep learning algorithms and architecture. To compare them multiple algorithms were generated and trained on benchmark databases and their training history was plotted to heatmaps. Further investigation was done based on the information found on the heat maps. All benchmark databases were taken from Keras's library and all architectures were also implemented in Keras library and ran in Jupyter notebook. Training environments were Google Colaboratory and my personal computer. Chosen databases were image classification databases and for comparing recurrent architecutres IMDB and Reuters database were chosen. Work resulted in the illustration of feed forward neural networks weaknesses on image classficiation, the tendency of overfitting with convolutional neural networks and popular methods for dealing with that issue. The ability of sentiment analisys and text classification for small GRU and LSTM models was illustrated and their performance differences with data padding alterations.

The thesis is in Estonian and contains 23 pages of text, 6 chapters, 41 figures, 0 tables.

Lühendite ja mõistete sõnastik

MNIST	Modified National Institute of Standards and Technology
CIFAR	Canadian Institute For Advanced Research)
IMDB	Internet Movie Database
GRU	Gated recurrent unit
LSTM	Long short-term memory

Sisukord

Jooniste loetelu

1 Sissejuhatus

Viimastel aastatel on süvaõppe algoritmid endale palju tähelepanu tõmmanud. Google avalikustas enda sisese tensorflow teegi ning see on omandanud endale suure kasutajaskonna. Süvaõppe algoritmid on arvutusvõimsuse ning tarkvaraliste tööriistade arengu tõttu esile toonud mitmeid silmapaistvaid saavutusi.

Käesolevas töös võrreldakse klassikaliste pärivoolu närivõrkude ning konvolutsiooniliste närvivõrkude võimekust ja puudujääke pildi klassifitseerimises ning rekurrentsete närvivõrkude GRU ja LSTM skaleeruvust ning võimekust sentimendi analüüsis IMDB arvustuste andmebaasil.

Töö eesmärk on illustreerida nende arhitektuuride erinevusi benchmark andmebaasidel treenimisel, mõõtmete mõju treenimisele, treenimisega kaasnevaid probleeme ja nende probleemide populaarsemate lahenduste efektiivsust.

2 Optimeerimise meetodid

Süvaõppe arhitektuuride optimeerimiseks kasutatakse mitmeid meetodeid, põhiliseks optimeerimis meetodiks on õpialgoritm ise, kuid selle puudujääkide kompenseerimiseks on välja arendatud mitmeid arhitektuurilisi lahendusi vastavalt arhitektuurile ning õpitava andmehulga karakteristikutele.

2.1 Gradientlaskumise optimeerimise meetodid

Gradientlaskumine on kõige levinum närvivõrkude optimeerimise algoritm. Olenevalt andmete omadustest võib standartne versioon osutuda mittepiisavaks. Selles peatükis tuuakse välja erinevad gradientlaskumise ja selle optimeerimise meetodid.

2.1.1 Plokk gradientlaskumine

Plokk gradientlaskumien on kõige lihtsam versioon, mis arvutab kulufunktsiooni kõigi parameetrite suhtes arvestades kogu treening admehulgaga :

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

Kuna gradientidel tehakse uuendus peale kogu andmehulga töötlust osutub antud meetod aeglaseks ning mälukulukaks [1].

2.1.2 Stohhastiline gradientlaskumine

Stohhastilisel gradientlaskumisel arvutatakse gradient suvaliselt valitud üksikute andmepunktide baasil:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

Üksikute punktide baasil vähendab ajalist kulu, kuid kuna andmepunktid valitakse suvaliselt ja gradient arvutatakse peale individuaalseid punkte võib gradient lokaalse optimumi poole liikudes rohkem „ringi hüpelda“, mis võib raskendada lokaalsesse optimumini jõudmist, kuid võib ka tekitada võimalusi muude lokaalsete optimumide leidmiseks [1].

2.1.3 Mini-plokk gradientlaskumine

Stohhastilisel gradientlaskumisel arvutatakse gradient suvaliselt valitud üksikute andmepunktide baasil:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

Selline lahendus vähendab parameetrite uuenduse varieeruvust ning võimaldab paljude teekide maatrikside optimiseerimisi ära kasutada. Enamustes lahendustes peetakse stohhastilise gradientlaskumise all silmas mini-plokk laskumist [1].

2.1.4 Inerts

Lokaalsete optimumide ümbruses on osade parameetrite gradiendid palju suuremad, kui teiste, mis võib põhjustada gradiendi hüplemist ümber optimumi. Selle probleemi lahenduseks on võetud kasutusele inerts. Inerts seisneb eelmise sammu uuendusvektori murru praegusele uuendusvektorile liitmises (ülehüpete puhul gradiendi märk muutub, mis tekitab liitmisel pidurdava efekti) [1].

2.1.5 Nesterovi kiirendatud gradient

Kui inertsi gradiendi arvutuse meetod pidurdas peale optimumist möödumist, siis Nesterovi kiirendatud gradient, muudab arvutatavat gradienti arvestades hinnangulisi tuleviku väärtusi:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$

$$\gamma - \text{inertsikordaja}, v - \text{uuendusvektor}$$

γv_{t-1} kujutab endast eelmise iteratsiooni inertsi, selle praeguse iteratsiooni parameetritest θ lahutamisel saab hinnangulise tuleviku iteratsiooni parameetrite väärtuse. Nesterovi kiirendatud gradiendi eeliseks on suurem tundlikkus [1].

2.1.6 Adagrad

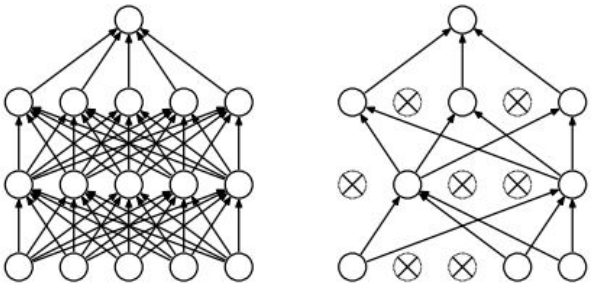
Kui varasemad algoritmid kasutasid kõikide parameetrite jaoks sama suurt õpisammu, siis adagrad uuendab õpisammu iga parameetri suhtes individuaalselt [1].
Adagradil on hulgaliselt edasiarendusi:

- Adadelta - üritab lahendada Adagradiga kaasnevat monotoonselt kahanevat õpisammu
- RMSprop - üritab lahendada sama mida adadelta
- Adam - Adadelta/RMSprop edasiarendus
- AdaMax - Adami edasiarendus
- Nadam – Adam Nestrovi kiirendatud gradiendiga

2.2 Arhitektuurilised optimeerimise meetodid

2.2.1 Dropout meetod

Dropout meetod seisneb treeningu käigus närvivõrgust neuronite eemaldamisel. Igast kihist eemaldatakse ajutiselt suvalised neuronid, koos nendesse sisenevate ning neist väljuvate ühendustega, vastavalt määratud dropouti protsendile(kui dropout on 25% siis igal antud kihi neuronil on 25% tõenäosus, et nad eemaldatakse ajutiselt võrgust), iga epohhi käigus eemaldatavad neuronid vahetuvad, validatsiooni ajal taastakse võrgust eemaldatud neuronid [2]. Dropouti tekitatud hõrendatud võrk vähendab neuronite sõltuvust individuaalsetest neuronitest, mis vähendab ka ülesobitavust. Hõrendatud närvivõrku saab näha joonisel 1.



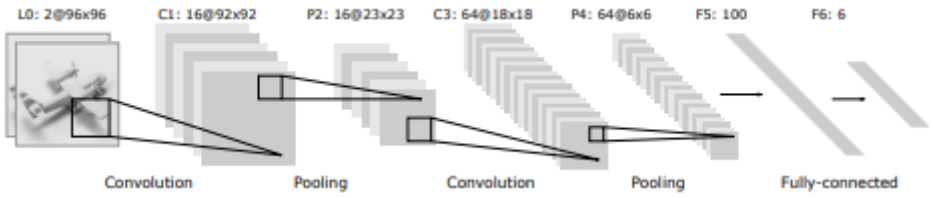
(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Joonis 1: Standartse(vasakul) ning dropoutiga(paremal) närvivõrgu mudel.

2.2.2 Ahenduskiht

Ahenduskihi eesmärgiks on tunnuse kaartide resolutsiooni vähendamisega ruumilise invariantsuse saavutamine, iga ahendatud kiht vastab kindlale konvolutsiooni omaduse kihile(Joonis 2) [3].



Joonis 2: Ahenduskihi rakendus konvolutsioonilisel närvivõrgul.

Max pooling ahenduskiht võtab iga nxn ruudu maksimumi, max pooling ahenduskihi samm on sama suur kui ta mõõtmed (üheltki pikslilt korduvalt maksimumi ei võeta).

3 Keskkonnad

3.1 Teegid

3.1.1 Tensorflow

TensorFlow on open-source teek andmevoo ja differenceeruva programmeerimise jaoks, mida kasutatakse masinõppe lahendustel. Tensorflow on arendatud Google Brain tiimi poolt Google siseseks kasutuseks.

3.1.2 Keras

Keras on open-source närvivõrgu teek pythonis. Seda saab jooksutada TensorFlow, Microsoft Cognitive Toolkit-i, Theano ja PlaidML peal. Keras fokuseerib kasutajasõbralikkusele, modulaarsusele ja laiendatavusele. Antud töös on närvivõrkude genereerimisel kasutusel Keras tensorflow baasil, kerase kasutamine võimaldas tensorflow teegi otsest kasutamist täielikult vältida.

3.2 Arenduskeskkonnad

3.2.1 Jupyter Notebook

Jupyter Notebook on weebipõhine interatkiivne arvutuskeskkond, kus on võimalik koodi jupiti jooksutada ja koodi juppide vahele teksti/pilte sisestada, mis võimaldab koodi juppe jooksvalt modifitseerida(ilma muutujate kadumiseta) ning koodi töö põhimõtte piltlikult kasutajale arusaadavaks teha.

3.2.2 Colaboratory

Colaboratory on tasuta Jupyter Notebook keskond, mida saab jooksutada pilves. Notebookid salvestatakse google drivele. Coldaboratory toetab Python 2 ja Python 3 programmeerimise keeli. Kuigi Jupyter Notebooki saab kasutada ka lokaalselt, otsustasin antud töös seda jooksutada colaboratorys kuna see lihtsustas sõltuvuste laadimist notebookidesse.

4 Andmebaasid

4.1 MNIST

MNIST (Modified National Institute of Standards and Technology) andmebaas on käsitsi kirjutatud numbrite andmebaas, mida kasutatakse tihti pildi töötlus süsteemide ja masinõppe algorimide treenimiseks. MNIST koosneb 60000 annoteeritud treening pildist ja 10000 testpildist. Pildid on mustvalged ühekohalised numbrid (0-9) ning piksli eredus on esitatud 256 erineval eredusel ning pildi mõõtmed on 28x28 pikslit.

4.2 CIFAR10

CIFAR10 (Canadian Institute For Advanced Research) andmebaas on piltide kollektsioon , mis nagu MNIST-ki on kasutusel pildi töötlus süsteemide ja masinõppe algorimide treenimiseks. CIFAR koosneb 60000-st 32x32 piksliga värvipildist, mis kuuluvad 10-sse erinevasse klassi(lennukid, autod, linnud, kassid, hirved, koerad, konnad, hopused, leavad ja veoautod).

4.3 CIFAR100

CIFAR 100 on sisuliselt sama, mis CIFAR10, klasse on 100, kuid andmehulk klassi kohta on väiksem, kogu andmehulk andmebaasis on 60000 (600 klassi koha) .

4.4 IMDB filmi arvustuste sentimentide analüüsi andmebaas

IMDB andmebaas on Kerase teegis saadaval olev imdb filmi arvustuste kogumik, mis on sildistatud positiivselt/negatiivselt (0/1). Andmebaas koosneb 25000-st kommentaarist, milles esinevad sõnad on esindatud numbritega sõnade populaarsuse järjekorras.

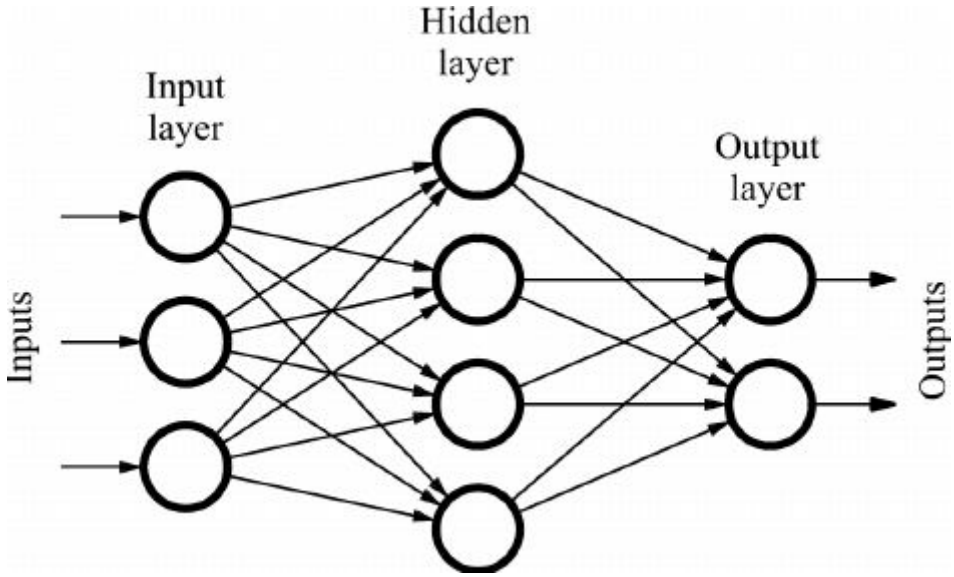
4.5 Reuters uudiste teemade klassifitseerimise andmebaas

Reuters uudiste teemade klassifitseerimise andmebaas on Kerase teegis saadav uudiste kogumik. Andmebaasis on 11228 uudist, mis on jaotatud 46 teema vahel. Sõnad on esindatud numbritega sõnade populaarsuse järjekorras.

5 Süvaõppe närvivõrkude võrdlus

5.1 Pärilevivõrgud

Mitme kihilised pärilevivõrgud(feed-forward neural network) koosnevad neuronitest, mis on jaotatud kihtidesse. Esimest kihti kutsutakse sisendkihiks, keskmist kihti kutsutakse peidetud kihiks ja viimast kihti kutsutakse väljundkihiks [4]. Sisendivektor korrutatakse läbi kaalumaatriksiga, mille väljundmaatriks kujutab iga esimese peidetud kihi neuroni jaoks vastavat sisendvektorit, sisendvektor koosneb kaaluga korrutatud sisendkihi elementidest, neuron summeerib sisendvektori ja korrutab summa aktivatsioonifunktsiooniga. Peidetud kihtide vahel toimub arvutus analoogselt, aga sisendkihi elemendid on vahetuses eelmise kihi neuronite aktivatsioonifunktsiooni väljunditega. Väljundkihi sisenditeks on viimase peidetud kihi väljundid. Väljundkihi neuronite arv on soovitud väljundite arv(Joonis 3). Antud töö kontekstis andmebaasides olevate piltide klasside arv.



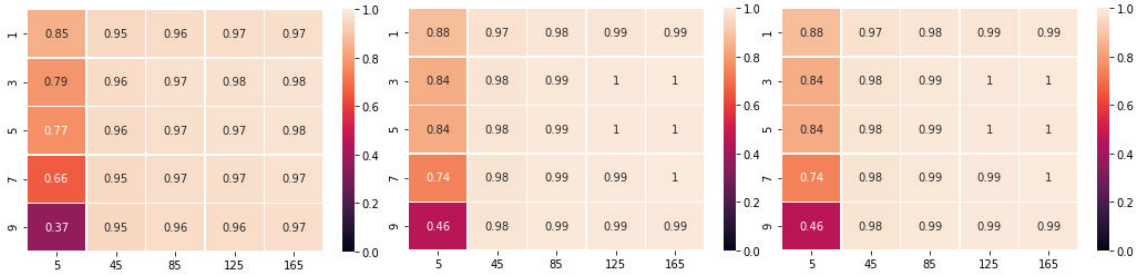
Joonis 3: Pärilevivõrgu mudel.

Pärilevivõrkude uurimiseks genereerisin erineva laiuse ning sügavusega võrgud, mida treenisin erinevate andmete peal, et saada üldine arusaam võrgu omadustest. Võrgu laiuseks valisin algselt 5 neuronit, ning suurendasin laiust 20 kaupa kuni 185 neuronini(MNIST andmebaasil 40 kaupa 5-165 neuronit). Peidetud kihtide arv algas 1-ga ja lõppes 10-ga , kõik peidetud kihid olid neuronite arvult identsed(20-185). Optimeerimiseks kasutasin AdaDelta optimeerijat. Kokku genereerisin 100 erinevate mõõtmetega võrku. Treenimise kestvuseks valisin 10 epohhi, näidiste arvuks(batch size) enne gradiendi uuendust valisin 128, ühe epohhi käigus läbitakse terve treening andmehulk 128 näidise kaupa. Pärilevivõrku treenisin MNIST, CIFAR10 ja CIFAR100 andmebaasidel. Treeningu käigus salvestasin iga võrgu jaoks selle treening vea, treening täpsuse, validatsiooni vea ning validatsiooni täpsuse, ning genereerisin vastad heatmapid, kus kujutasin keskmist viga, viimast viga, keskmist täpsust, viimast täpsust nii treening kui validatsiooni andmete jaoks. Validatsiooni täpsuse jaoks tegin ka parima tulemuse heatmapi. Keskmise ja viimase tulemuse jaoks tegin eraldi heatmapid, et oleks

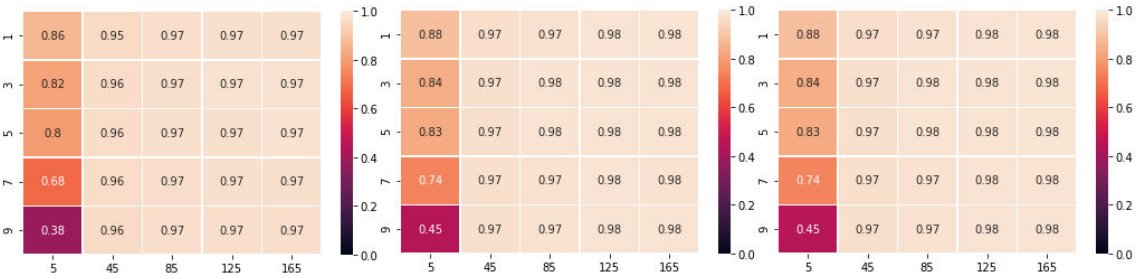
võimalik vea koonduvust hinnata. Pärilevivõrkudes peetakse „kihtide arvu“ all antud töös silmas peidetud kihtide arvu k.a. sisend kiht.

5.1.1 MNIST-l treenimine

MNIST andmebaasil Treenimisel oli parimaks treening tulemuseks üle 99,5%, parim validatsiooni tulemus oli 98%, pärilevivõrgul oli raskusi treenimisel kitsastes ning sügavates võrkudes(joonis 4 - 5).



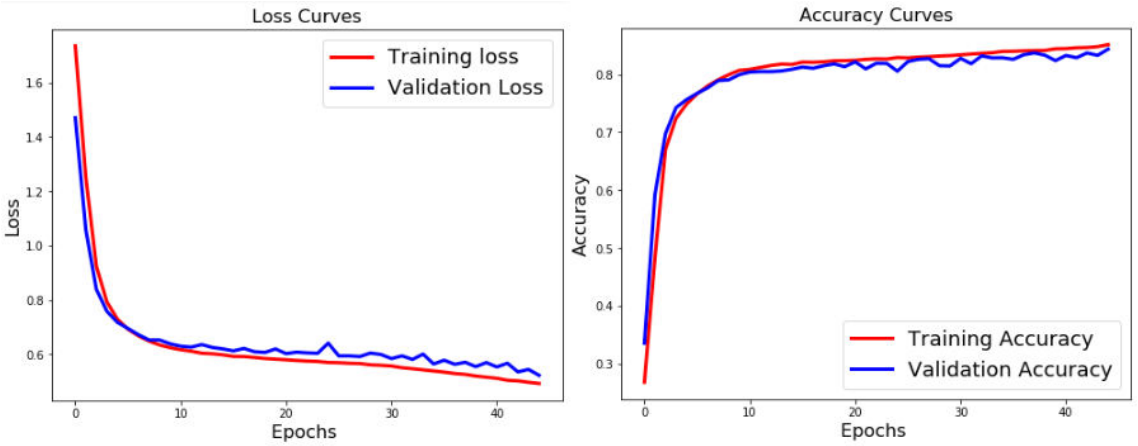
Joonis 4: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).



Joonis 5: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

Kitsastes ja sügavates võrkudes erinesid validatsiooni tulemused treening tulemustest 1% võrra, mis viitab et pikemal treenimisel oleks võimalik treenimise tulemusi parandada.

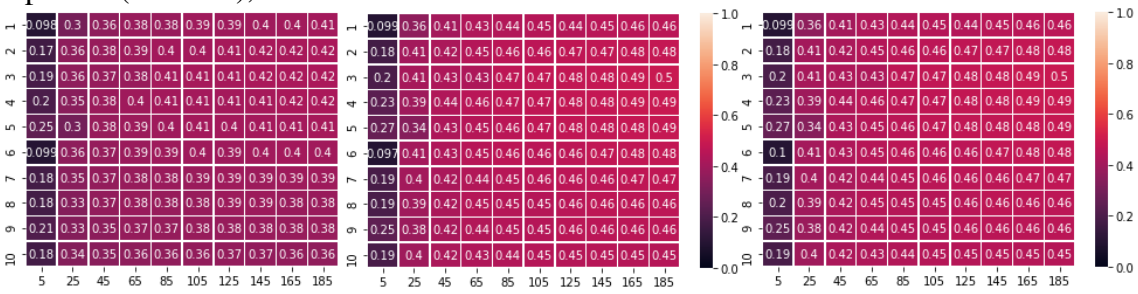
5 neuroni laiuse ning 8 kihi sügavune pärilevivõrk treeniti 45 epohhi ning parim validatsiooni tulemus kasvas 84%-ni, mis on madalam 1 kihiga võrgu 88%-st. Treenimise lõpul oli täpsuse trend aeglaselt kasvamas(Joonis 6), kuid antud katse viitab, et MNIST andmebaasil on treeningu tulemus suuremas osas piiratud esimese kihi laiusega.



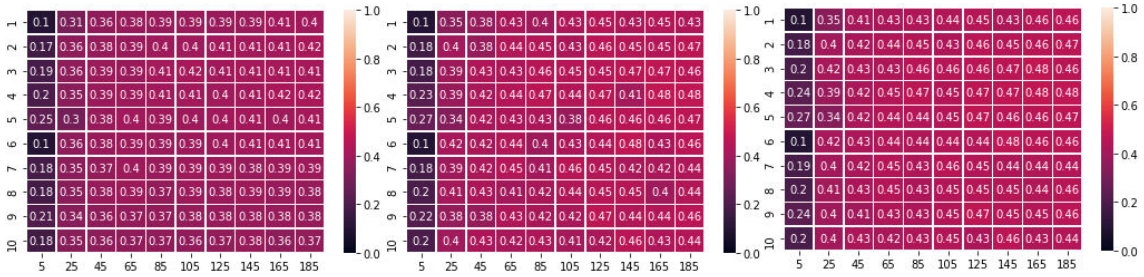
Joonis 6: 5x9 mõõtmetega pärilevi võrgu treenimise tulemused 45 epohhil.

5.1.2 CIFAR10-l treenimine

CIFAR10-l treenimisel oli parimaks validatsiooni täpsuseks 48%. Üldjuhul validatsiooni täpsused jäid paari protsendi kaugusele treening täsustest, mis viitab, et ülesobitavus ei olnud madala validatsiooni täpsuse põhjustajaks. 5 neuroni laiused võrgud olid kõik nõrkade tulemustega (10%-25%), võrgu töövõimekus paranes kuni viienda peidetud kihi lisamiseni sellel laiusel(Joonis 8). Võrgu laiendamisel paranesid tulemused igal sügavusel, leidus mõningaid erandlikke andmepunkte eri sügavustel mis sellele mustrile ei vasta, kuid nende hõredus ning juhuslik paigutus viitab juhuslikule ebasoodsale algväärtustamisele. Parimad tulemused validatsiooni andmetel paiknesid esimesest viienda kihini, edasine kihtide lisamine vähendas validatsiooni tulemusi. Selles vahemikus paiknesid tulemused 46%-48% vahemikus. Kihtide lisamine ei andnud märkimisväärselt paremaid tulemusi välja arvatud väga kitsastel võrkudel. Kuna treening kestis ainult 10 epohhi ning treening täpsus oli väga lähedal validatsiooni täpsusele(Joonis 7), mis viitab et treenida tuleks kauem.

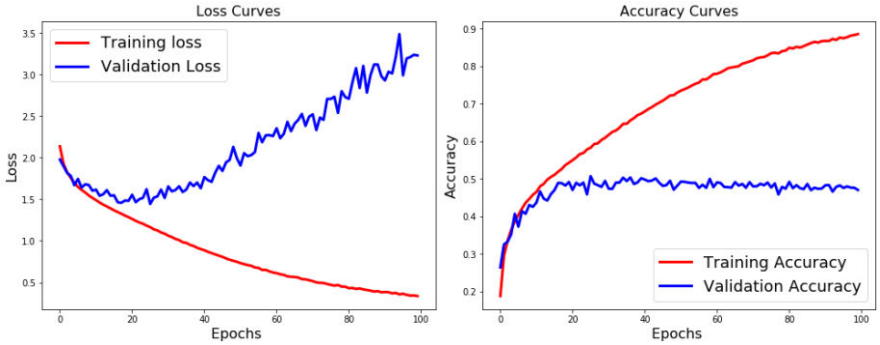


Joonis 7: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).



Joonis 8: Validatsiooni täpsuse heatmapid(keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

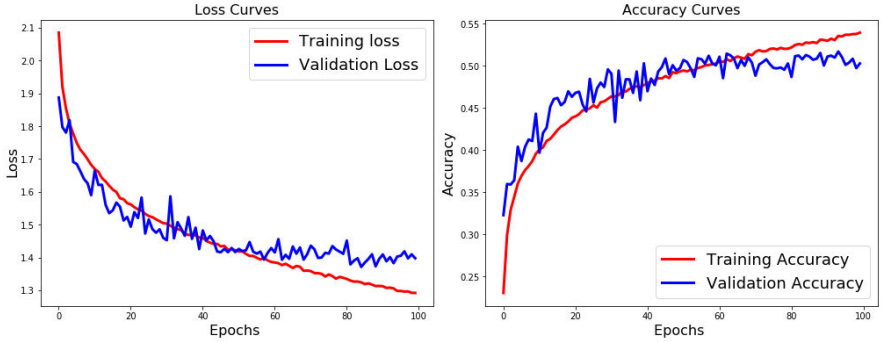
Üheksa kihi sügavuse ning 185 neuroni laiuse närvivõrgu 100 epohhisel treenimisel(Joonis 9) kerkis maksimaalne treening täpsus 88%-ni, kuid test andmetel tõusis see vaid 50%-ni, mis viitab ülesobitavusele. Sama sai katsetatud ka muude kihtide arvudega ning ainus erinevus seisnes ülesobitavuse kiiruses.



Joonis 9: 9x185 närvivõrgu 100 epohhilise treenimise tulemus.

Kuigi heatmapid viitavad, et võrgu laiendamine võib tulemusi parandada, jäi närvivõrgu laiendamisel 365 ja 2005 neuronini validatsiooni täpsus alla 52%.

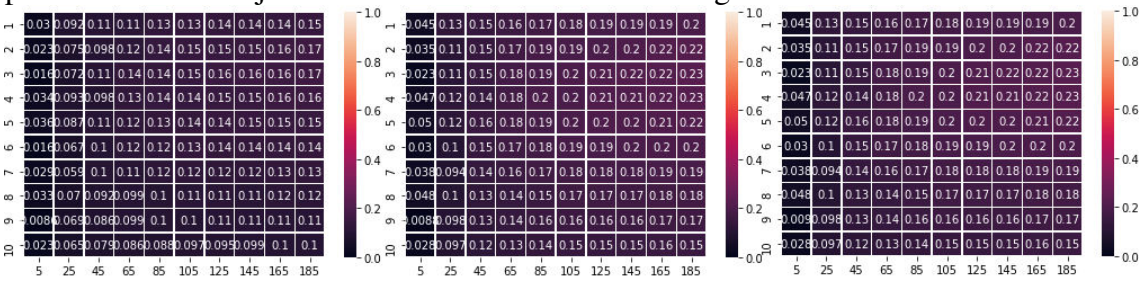
Dropouti kihi (25%) lisamisel vähenes ülesobitavus märgatavalt, 185 neuroni laiuse ning 2 varjatud kihi sügavune närvivõrk 100 epohhi treenimisel(Joonis 10) hoidis täpsuse treening ja validatsiooni andmetel lähestiku 60 epohhini, peale mida võrk hakkas jällegi ülesobitama. Validatsiooni täpsus jäi jätkuvalt alla 52%. Kihi lisamisel ülesobitavus vähenes, kuid võrreldes varasemaga validatsiooni tulemus ei paranenud. Adadelta optimeerija Adami optimeerija vastu välja vahetamine vähedas validatsiooni täpsuse 51% juurest 45% peale.



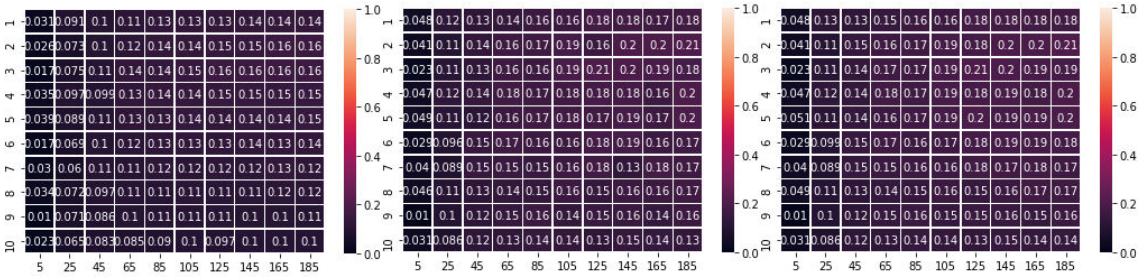
Joonis 10: 25% dropoutiga 9x185 närvivõrk.

5.1.3 CIFAR100-l treenimine

Nagu CIFAR10 puhul, on ka CIFAR100 puhul väga kitsaste ja sügavate võrkude treeningu ja validatsiooni täpsused väga madalad. Viie neuroni laiustel võrkudel oli maksimaalne täpsus 1-5% vahemikus(joonis 11-12), kusjuures kihtide lisamine ei tekitanud ilmselget trendi ei täpsuse kasvamise, ega kahanemise suunas, mis viitab treeningu tulemuse tugevale sõltumisele võrgu juhuslikule algväärtusele. Võrgu laienemisel paranes treening andmetel täpsus sujuvalt sarnaselt CIFAR10 treeningule, treeningu parimad tulemused jäid 185 neuroni laiuse ning 2-5 kihi sügavustesse võrkudesse (20% - 23%). Validatsiooni andmetel oli maksimaalne täpsus 21% ning parimad tulemused jäid samasse vahemikku kui treening andmetel.

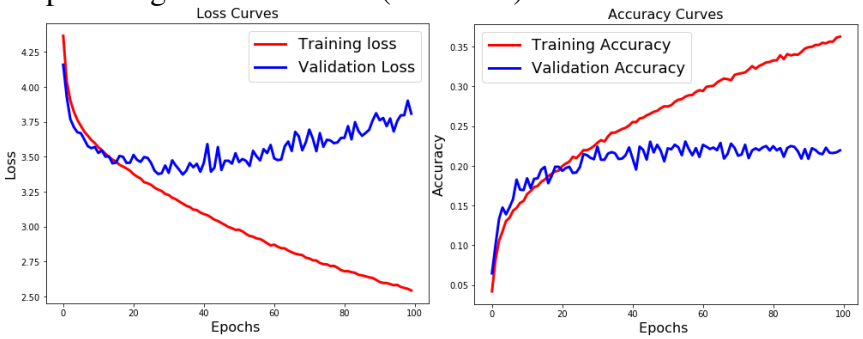


Joonis 11: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal) .



Joonis 12: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

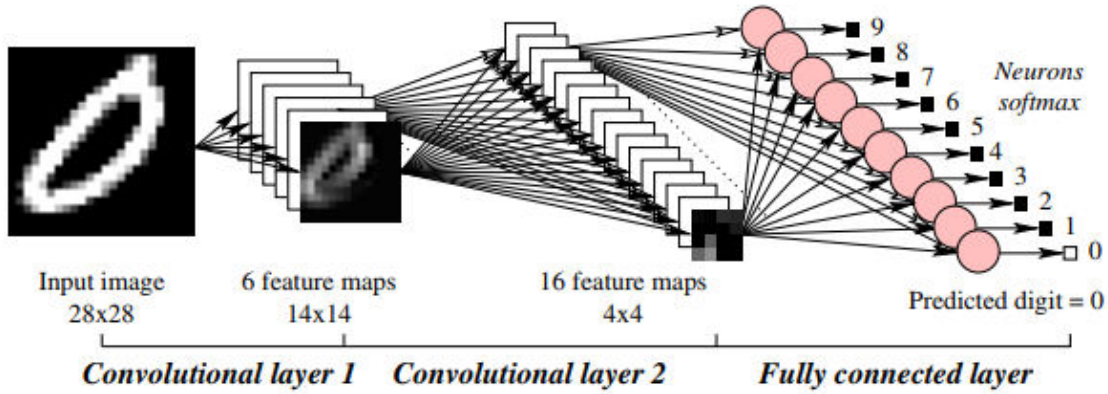
Täpsus kasvas üldjoones võrgu laienemise suunas, peale 365 neuroni laiuse võrgu koos dropout kihiga(25%) 100 epohhi treenimist oli parim validatsiooni tulemus oli 23% , peale 20-dat epohhi algas ülesobitamine(Joonis 13).



Joonis 13: 25% dropout 1x365 pärilevivõrk 100 epohhi treenimise tulemus.

5.2 Konvolutsioonilised võrgud

Konvolutsioonilisi võrke kasutatakse kahemõõtmelistele massiividele närvivõrkude rakendamiseks(üldjuhul pildid). Konvolutsioonilise võrgus toimub pildi kaardistamine läbi konvolutsioonide. Iga neuron rakendab endale spetsiifilist filtrit igale sisendregioonile(Joonis 14).

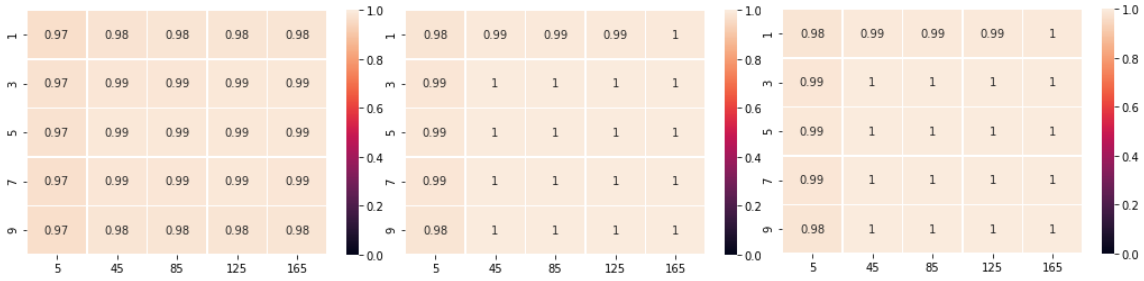


Joonis 14: Konvolutsiooniline närvivõrk.

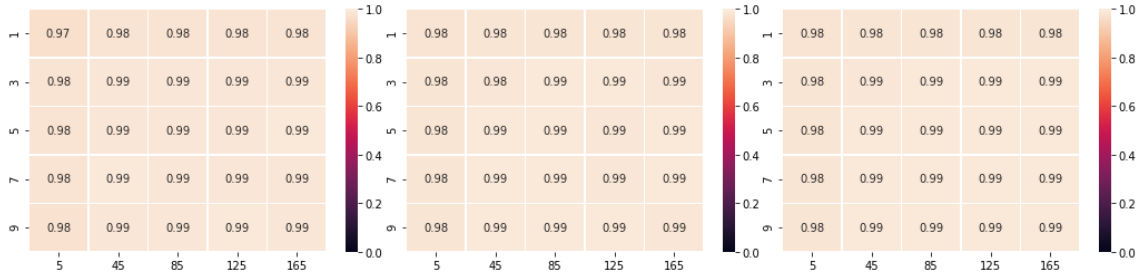
Antud lähenemine on pilditöötluses tugev, kuna kõrge sisendidimensionaalsus võib piiranguteta süsteemis treenimise raskusi tekitada [5]. Kuna üldjuhul on kasutusel rohkem kui 1x1 mõõtmeline konvolutsiooni filter, siis iga konvolutsiooni kihi järel väljundite mõõtmed vähenevad. See omadus tekitab võrgul võimaluse uurida laiemaid regioone.
Konvolutsiooniliste närvivõrkude uurimiseks genereeriti 10 erineva sügavusega närvivõrku(MNIST puhul oli samm 40, laius lõppes 165-ga, sügavust suurendati 2 kihi haaval). Konvolutsiooni filtrite arvud olid kihiti samad ning filtrite arv algas 5-st ja lõppes 185-ga, filtrite arvu kasv kihi kohta suurenes 20 kaupa. Võrke treeniti 10 epohhi, näidiste arv oli 128. Konvolutsiooni filtrite suuruseks valisin (3x3).

5.2.1 MNIST-l treenimine

Kõik validatsiooni tulemused MNIST andmebaasil treenides ületasid 98%, treening ja validatsiooni tulemused teineteisest tugevalt ei erinenud (Joonis 15-16). Võrreldes pärilevivõrguga oli 5 konvolutsiooni kihiline võrk palju võimekam, kuid konvolutsioonilise kihi väljundiks on identne kiht pärilevivõrgu kihtidega ning kihi laius on 10(validatsiooni klasside arv).

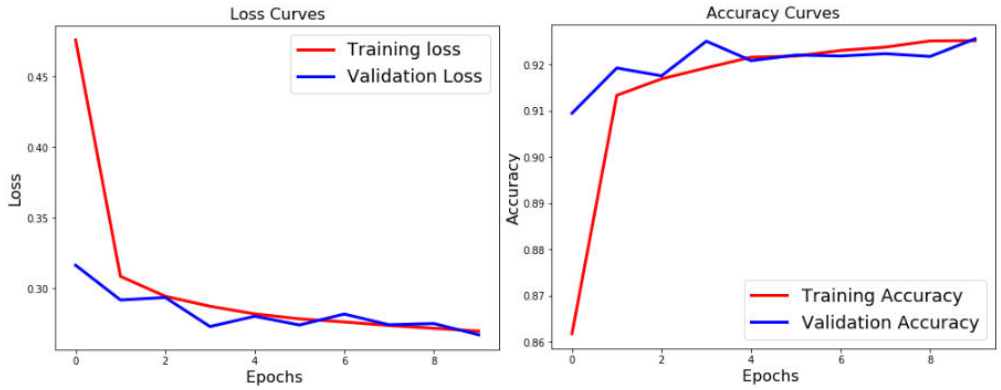


Joonis 15: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).



Joonis 16: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

Konvolutsioonide mõju võrdlemiseks pärilevivõrkude peidetud kihiga genereeriti 1 konvolutsiooniga mudel, mida treeniti 10 epohhi(Joonis 17). Parimaks validatsiooni täpsuseks oli 92%, mis oli vähem 5 konvolutsiooniga 98%-st mis viitab, et konvolutsiooni kihtide lisamisel eksisteerib mõju võrgu treenimisele.

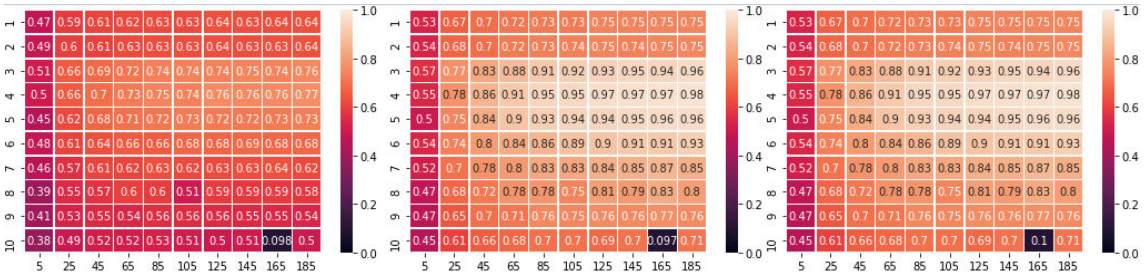


Joonis 17: 1 konvolutsioonilise kihiga võrgu 10 epohhilise treenimise tulemus.

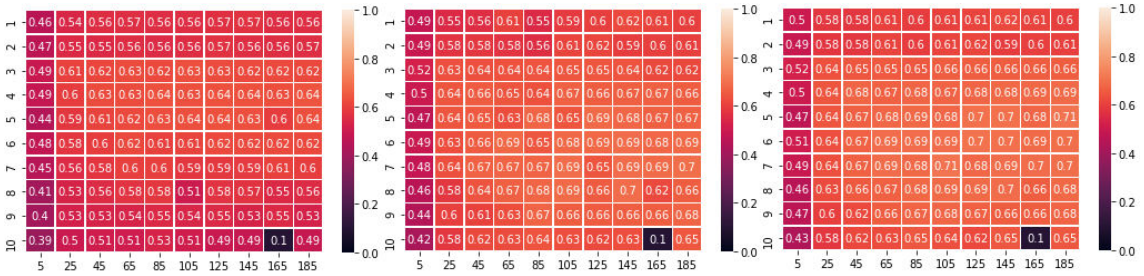
5.2.2 CIFAR10-l treenimine

CIFAR10-l treenimisel oli parimaks klassifikatsiooni täpsuseks validatsiooni andmetel 71%. Treening andmetel oli suurimaks täpsuseks 98%, mis viitab tugevale ülesobitamisele. Kõige tugevam oli ülesobitamine 3 kuni 8 kihi sügavustes konvolutsioonilistes võrkudes(Joonis 18). Filtrite lisamisel nii võrgu test kui validatsiooni täpsus kasvasid. Selles võrkude hulgas tulid esile ka kõige paremad

validatsiooni tulemused. Kõige intensiivsem ülesobitamine oli 3-5 kihilisetes võrkudes ning 70%st paremad tulemused valiatsiooni andmetel jäid 5-8 kihilistesse võrkudesse(Joonis 19).

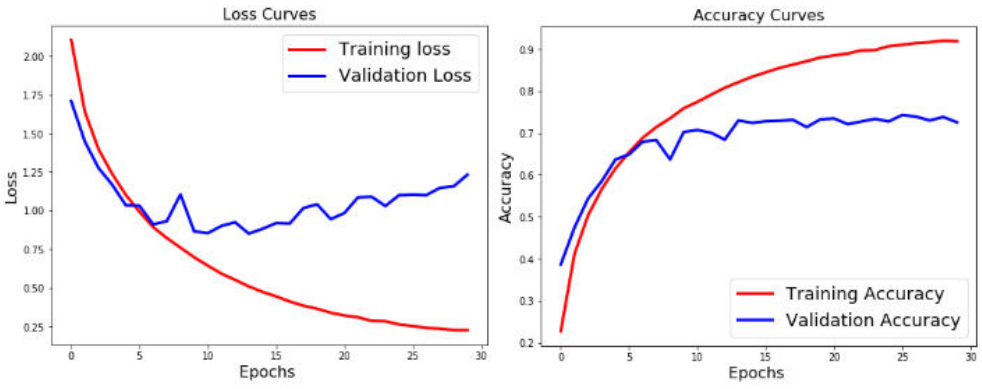


Joonis 18: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

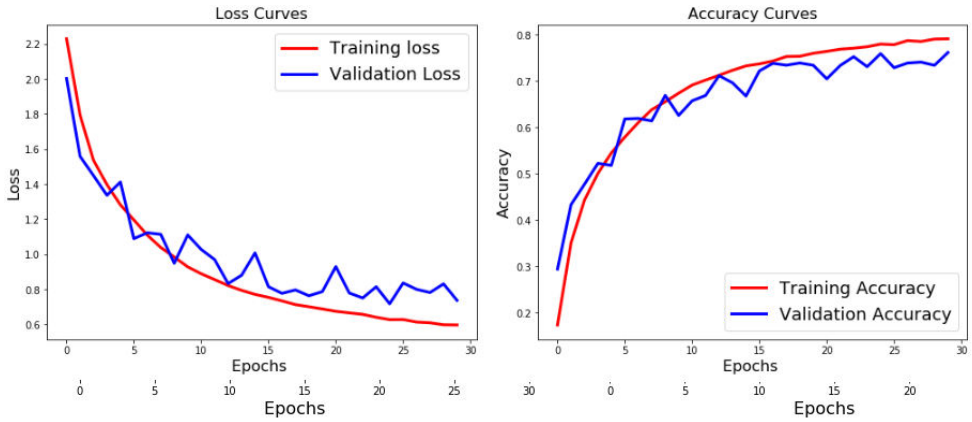


Joonis 19: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

Kuna ülesobitamine oli madalam 5-8 kihilistes võrkudes, treeniti 7 kihilist 105 konvolutsioonilise filtriga võrku 30 epohhi nägemaks kui palju on võimalik validatsiooni tulemust tõsta. Parim tulemus katsel oli 67%, mis saavutati 7-l epohhil(Joonis 20), sellele järgnevalt hakkas võrk ülesobitama ning validatsiooni täpsus jäi 66% ümbrusesse. Dropouti (25%) lisamine igale kihile peale esimese tõstis 30 epohhilisel treenimisel validatsiooni andmetel 74% täpsuse, treening andmetel tõusis täpsus 92%-ni, mis tähendab, et ülesobitamine oli küll vähendatud , kuid siiski probleem. Dropouti suurendamine 40%-ni viis parima validatsiooni täpsuse 76%-ni, treening andmetel kasvas täpsus 84%-ni(Joonis 21).



Joonis 20: 7x105 konvolutsioonilise võrgu 30 epohhi treenimise tulemus.



Joonis 21: 1x105 40% konvolutsioonilise võrgu 30 epohhi treenimise tulemus.

Dropouti suurendamine 50%-ni tõstis täpsuse samuti 76%-ni, kuid treening täpsus oli 79%. Lisa 10 epohhi treenimisel tõusis validatsiooni täpsus 0.36%, treeningandmetel oli tõus 1.5%, mis viitab aeglasemale kuid siiski eksisteerivale ülesobitamisele.

Edasise tulemuse parandamiseks rakendati ahenduskihti(max pooling). Filtri suuruseks valiti 2x2, filter sai lisatud nagu dropoutki kõikidele kihtidele peale esimese. Kuna ahendus kihid vähendavad väljundi suurust(Joonis 22) , ei olnud 7 kihilisele võrgule iga kihilist ahendust võimalik rakendada. Uueks võrguks valiti sama laia, kuid 4 kihilise konvolutsioonilise võrgu, mille viimasele 3-le kihile rakendasin nii ahendust kui 50% dropouti.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 30, 30, 105)	2940
conv2d_2 (Conv2D)	(None, 28, 28, 105)	99330
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 105)	0
dropout_1 (Dropout)	(None, 14, 14, 105)	0
conv2d_3 (Conv2D)	(None, 12, 12, 105)	99330
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 105)	0
dropout_2 (Dropout)	(None, 6, 6, 105)	0
conv2d_4 (Conv2D)	(None, 4, 4, 105)	99330
max_pooling2d_3 (MaxPooling2	(None, 2, 2, 105)	0
dropout_3 (Dropout)	(None, 2, 2, 105)	0
flatten_1 (Flatten)	(None, 420)	0
dense_1 (Dense)	(None, 10)	4210

Total params: 305,140
Trainable params: 305,140
Non-trainable params: 0

Joonis 22: Konvolutsioonilinse ahenduskihiga närvivõrgu struktuur.

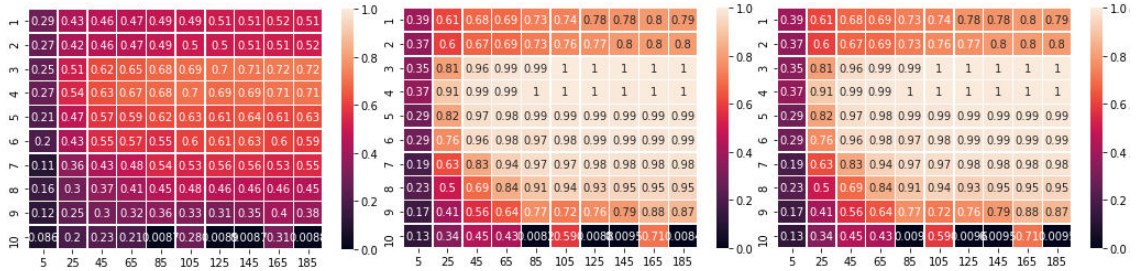
5.2.3 CIFAR100-l treenimine

Konvolutsiooni filtrite poolt tekitatud kujutiste mõõtmete ning kujutiste arvu korrutis on väljundi kihi suhtes sisend parameetrite arv, mis on igale väljundile kaaluga parametriseeritud sisendiks. Kui MNIST ja CIFAR10 puhul tõusis parameetrite arv mõne miljonini, siis CIFAR100 puhul tekkis olukordi, kus viimane kiht omas 96%(Joonis 23) kogu võrgu treenitavatest parameetritest. Antud kontekstis, kus filtrite kogus varieerus 5 ja 185 vahel ja võrgu sügavus varieerus 1-10 konvolutsiooni kihi vahel, tekitas ühe kihiline 185 filtriga konvolutsiooniline võrk väljundisse 30*30*185*100 = 16 650 000 treenitavat parameetrit.

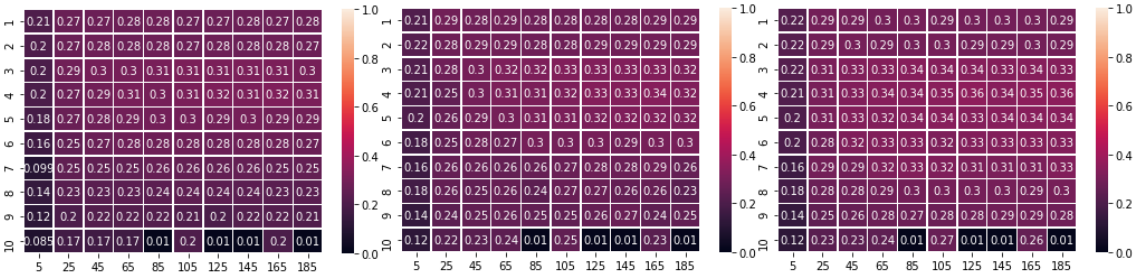
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 145)       4060
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 145)       189370
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 26, 26, 145)       189370
_________________________________________________________________
flatten_1 (Flatten)          (None, 98020)             0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               9802100
=================================================================
Total params: 10,184,900
Trainable params: 10,184,900
Non-trainable params: 0
```

Joonis 23: Kolme kihiline 10184900 parameetriline konvolutsiooniline võrk.

Validatsiooni heatmap treeningandmetel näeb sarnane välja CIFAR10-l treenitud tulemustega, samuti on kõige suurem ülesobitumine 3-5 kihi sügavustel võrkudel(Joonis 24), filtrite lisamisel ülesobitumine kasvas. Kuna CIFAR100 on 10 korda väiksema andmehulgaga(klassi kohta) ja 10 suurema klasside hulgaga kui CIFAR10 on sellel treenimine märgatavalt raskendatud. Validatsiooni andmetel oli kõige täpsem tulemus 36%. Parimad klassifikatsiooni tulemused jäid samuti 3-5 kihi sügavustesse võrkudesse(Joonis 25). Kuna ülesobitamine on tugev pea iga mõõtmega konvolutsioonilisel närvivõrgul ei ole tõenäoliselt ühegi mudeli ilma muudatusteta pikem treenimine tulemuste rikas.

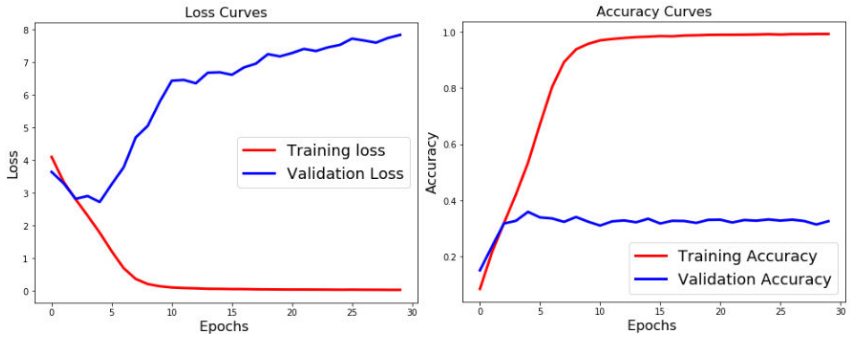


Joonis 24: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).



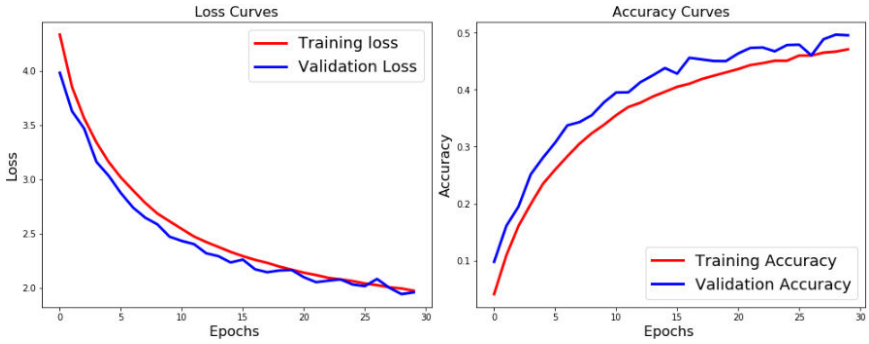
Joonis 25: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

Lisasin dropouti(25%, 40%, 50%, 70% ja 85%) 4 neuroni sügavusele 125 konvolutsiooni filtri laiusele võrgule. Peale 30 epohhi treenimist oli 25% dropoutiga mudeli treening andmetel täpsus 99% ja validatsiooni andmetel oli maksimaalne tulemus 35%(Joonis 26), 40% dropout andis samasuguse tulemuse ning treening andmete täpsus tõusis 97%-ni, 50% dropouti validatsiooni täpsus tõusis 37%-ni, test täpsus langes 95%-ni. Ülesobitamine algas kõigil dropoutidel 4-dast epohhist.



Joonis 26: 25% dropoutiga 4x125 konvolutsioonilise võrgu 30 epohhi treenimise tulemused.

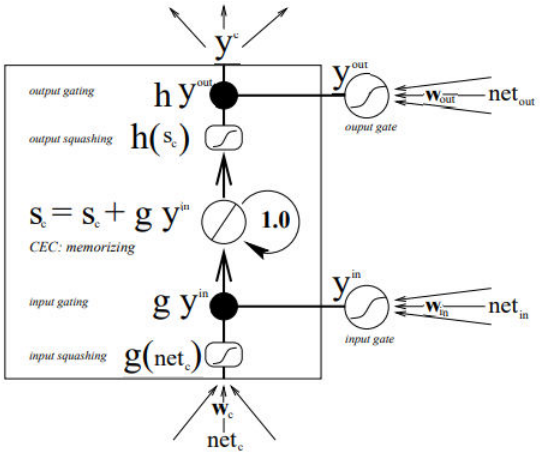
70%-ne dropout lükkas ülesobitavuse 5-6 epohhini, validatsiooni tulemused paranesid erinevalt varasematest katsetest kuni 21 epohhini. Maksimaalne validatsiooni täpsus oli 41%, treening täpsus tõusis 61%-ni. 85%-sel Dropoutil 30-ks epohhiks olid nii treening kui validatsiooni täpsused 31%, õppimise kiirus vähenes, kuid hoidis aeglaselt kasvavat trendi, 40 lisa ephohhil tõusis validatsioon 35%-ni.
Ahenduskihi rakendamisel mudelile sai sarnase mudeli CIFAR10 ahenduskihtidega võrgule erinevusega, et väljundkihis oli 100 neuronit. Dropoutiks sai valitud 40%, esimese 30 epohhiga oli parim validatsiooni täpsus 49%, treening täpsus kasvas 47%-ni, lisa 20 epohhi vältel tuli parimaks validatsiooni täpsuseks 51%, treeningtäpsus kasvas 50%-ni.



Joonis 27: Ahenduskihiga 4x125 konvolutsioonilise võrgu 30 epohhi treenimise tulemused.

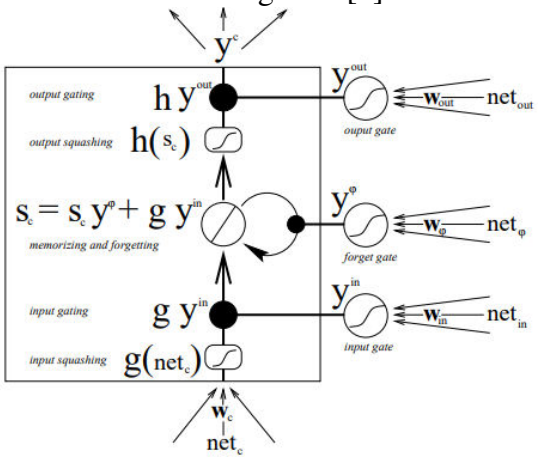
5.3 Long Short-Term Memory Unit

LSTM põhiline üksus varjatud kihis on mälu plokk(Joonis 28), mis koosneb ühest või rohkemast mälu elemendist ning paarist kohanduvast korrutavast üksusest, mis vahedavad siendeid ja väljuneid üksteise vahel plokis. Mälu elementide tuumaks on CEC(Constant Error Carousel), mille aktivatsiooni tõlgendatakse kui elemendi olekut. Uue sisendi või errori signaali puudumise korral jääb CEC lokaalne viga muutmata, mis lahedab kaduva vea probleemi[4].



Joonis 28:LSTM üksus.

Sisend ja väljund väravad reguleerivad antud mälu elemendi oleku(S_c) kirjutamist ja lugemist, net_c kujutab ploki sisendit, net_{in} kujutab input gate sisendit ja net_{out} kujutab output gate sisendit. Kuna mälu element võimaldab teatud olekut hoida pikka aega, siis eksisteerivad juhud kus mälu elemendi olek kasvab piiramatult (näiteks aja seeria info sisestamisel), mis võib väljundi küllastumist põhjustada, küllastumine kaotab väljundi tuletise, mis tähendab, et vea funktsioon ei mõju enam elemendile ning väljund võrdsustub väljundi sisendiga. Probleemi lahendab forget gate(net_φ)(Joonis 29), mis resetib mälu elemendi, kui nende sisu on aegunud [6].

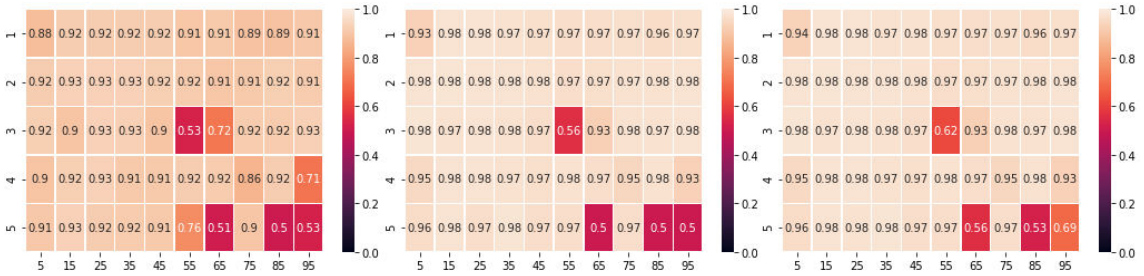


Joonis 29:LSTM üksus forget gate-ga.

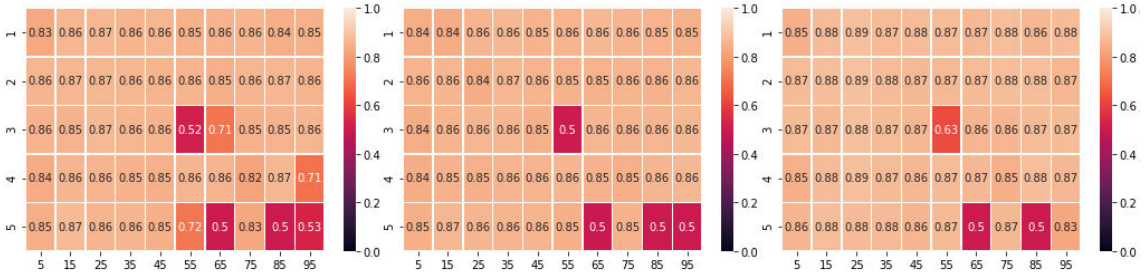
5.3.1 IMDB andmebaasil treenimine

Treenimiseks laeti alla kerase pakutud imdb library, arvesse võeti 10000 populaarseimat sõna. Treening andmed jaotati 960 sõnalistesse plokkidesse. Plokid kus sõnu oli vähem kui ette näthtud täitsin puuduvad sõnad 0-dega, 0 täidis paigutati plokkide ette otsa, et nad ei mõjutaks võrgu sisu. Võrgu sisendkihiks oli 8 kohaline manustamise(embedding) kiht, mis muutis andmetes olevad sõnad täisarvu kujult 8 kohalisteks murdarvu vektoriteks[8]. Andmehulgast 5% olid validatsiooni andmeteks. Võrgu laiuseks valiti 5-95 plokki, mida katsetati 10 plokiste vahedega, võrgu sügavused olid 1-5 kihti, treening mahuks oli 10 epohhi, optimiseeria oli Adam.
Treening tulemused nii treening kui validatsiooni andmetel ei varieerunud palju. Keskmise treeningu täpsuse heatmap näitab hõredalt treenimise raskusi sügavamate ja

laiemate kihtide hulgas(Joonis 30), 4 LSTM võrku olid keskmise treening täpsusega alla 54%. Treenimise lõpuks oli järgi 2 50% täpsusega kihti 5 kihi sügavustes võrkudes(Joonis 31). Probleemsete võrkude hõre paigutus viitab kaalude algväärtustamisega mitte vedamisele. Kõik võrgud peale 1 kihilise ja 5 ploki laiuse ning algväärtustamise probleemidega võrkude saavutasid treening täpsuse üle 94% ning validatsiooni täpsuse üle 85%. Maksimaalne validatsiooni täpsus oli 89%. IMDB andmebaasil treenimise kontekstis ei mõjutanud LSTM võrgu dimensioonid treeningu tulemusi.



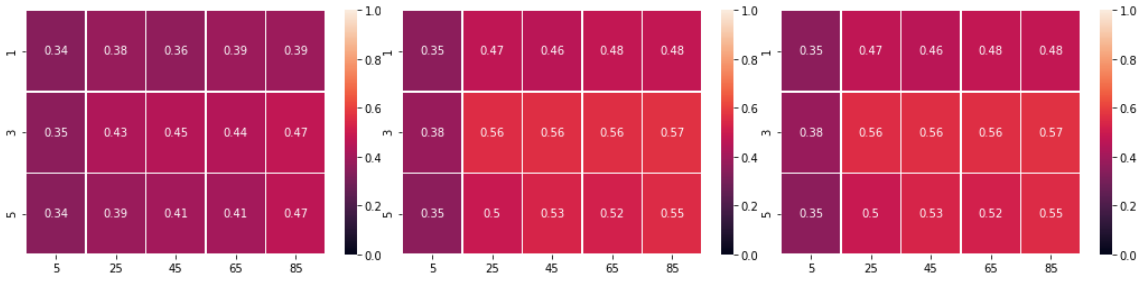
Joonis 30: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).



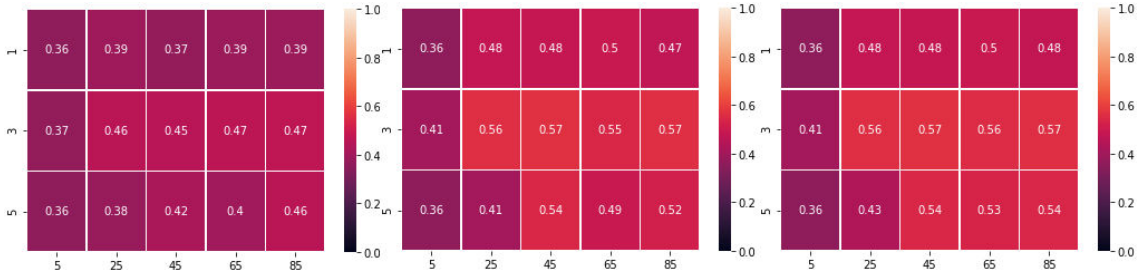
Joonis 31: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

5.3.2 Reuters andmebaasil treenimine

Reuters andmebaasil treenimisel treeniti LSTM mudeleid 8982 artiklil ning valideeriti 2246 artiklil. Parim validatsiooni tulemus oli 57%, treening täpsus oli samuti 57%. Võrgu parimad tulemused esinesid 2 kihilistes LSTM võrkudes. Kõige nõrgemad treenimise tulemused olid kitsastes ning õhukestes LSTM võrkudes(Joonis 32-33).

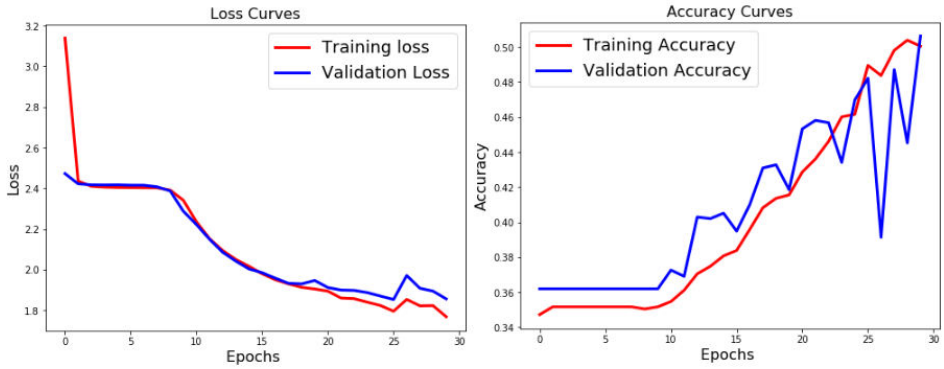


Joonis 32: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).



Joonis 33: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

Kõige õhemates kihtides jäi treening ning validatsiooni täpsuse vahe 2% sisse, mis viitab võrgu aeglasemale treenimisele. Peale 1 kihilise 5 plokkiga võrgu 30 epohhilist treenimist tõusid treening ning validatsiooni täpsused 50%-ni. Mudel oli esimesed 9 epohhi kinni 36%-se validatsiooni täpsuse peal(Joonis 34), mis viitab et esimesel epohhil leiti mingite kindlate sõnade ja klasside tugev korrelatsioon. Mudeli kinni jäämine võis olla tingitud andmebaasi väikesest mahust. Mudeli lisa 30 epohhi treenimisel tõusis validatsiooni täpsus maksimaalselt 54%-ni (53%-ni 3-dal epohhil), mis viitab, et antud mudeli limiit asub sama tulemuse läheduses. 1 ja 3 kihilise 85 ploki laiuse 30 epohhi treenimise parim tulemus oli 59% ja 60% vastavalt, 30 lisa epohhi treenimisel kasvas validatsiooni täpsus 62%-ni, kuid treeningtäpsus 70-ni. mis viitab, et kuigi mudelile kihtide lisamine kiirendas õppimist, ei mõjutanud see mudeli maksimaalset õppimisvõimet. Nullise täidise andmete lõppu panemine aeglustas LSTM treeningud tugevalt, kuid 1x85 plokise mudeli 60 epohhise treeningu lõpuks oli parim täpsus 48%, mis on võrreldav null täidise andmete ette panemise 10 epohhise tulemusega.



Joonis 34: 1x5 LSTM 30 epohhi treenimise tulemus.

36% oli keskmiste valideerimise tulemuste tabelis samuti väga tihti esindatud ning genereeritud treening andmeid lähemalt uurides leidsin, et enamus mudeleid jõudis selle tulemuseni esimesel validatsiooni epohhil. Valideerimise täpsus 0.36197686 esines andmetes 84 korral.

5.4 Gated Recurrent Unit

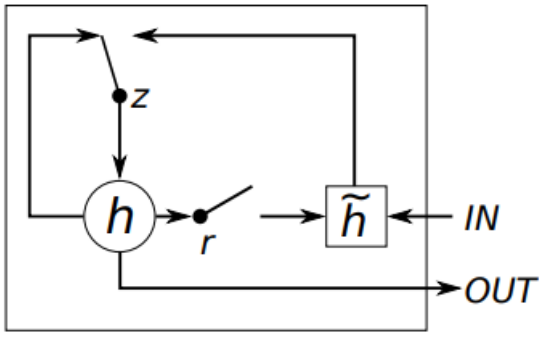
Sarnaselt LSTM-le on GRU-s üksuste vahelist info voogu kontrollivad väravad (gate) kuid GRU-s puuduvad eraldi mälu elemendid(Joonis 35). Aktivatsioon h_t^j ajaühikul t on lineaarne interpolatsioon eelmise aktivatsiooni h_{t-1}^j ja kandidaat aktivatsiooni $\tilde{h}_t^j$ vahel.

$$\mathrm{h}_t^j = (1 - \mathrm{z}_t^j)\mathrm{h}_t^j - 1 + \mathrm{z}_t^j \tilde{\mathrm{h}}_j^t ,$$

kus update gate z_t^j otsustab kui palju üksus enda aktivatsiooni või sisu uuendab.

$$z_t^j = \sigma(W_{x_t} + U_r h_{t-1})^j$$

σ on logistiline sigmoid. Eksisteeriva oleku ja uue oleku vahelise lineaarse summa võtmine sarnaneb LSTM-le, kuid GRU-s puudub mehhanism kontrollimaks kui palju ta sisene olek välissüsteemile avatud on [7].

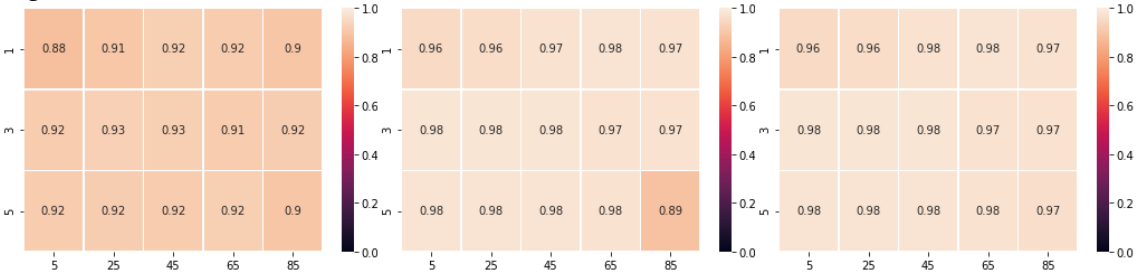


Joonis 35:GRU plokk.

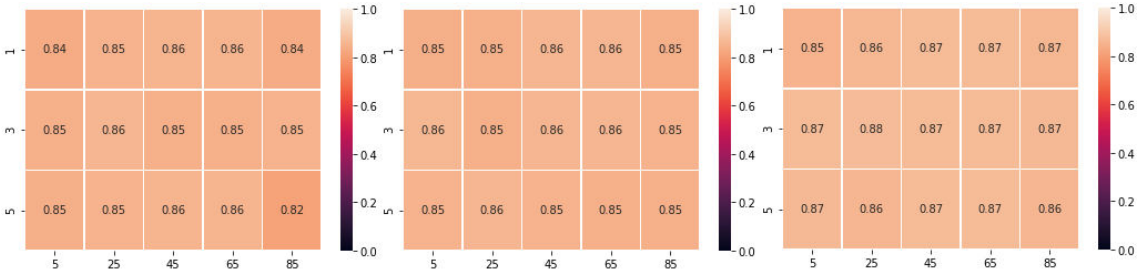
5.4.1 IMDB andmebaasil treenimine

IMDB andmete ettevalmistus GRU jaoks oli identne LSTM-le andmete ettevalmistusele. GRU jaoks genereeriti närvivõrke hõredamalt(20 kaupa ning sügavus

suurenes 2 kaupa).Ainus erinevus seisnes GRU plokkide kasutamises LSTM Plokkide asemel.
Seekord genereeriti 20-se sammuga 5-85 ploki laiune ning 2-se sammuga 1-5 kihi sügavune GRU närvivõrk. Parim Validatsiooni tulemus oli 87%, parim treening täpsus oli 98%(Joonis 36-37). Tulemused olid väga üksluised, kihtide ning GRU plokkide arvu muutmine ei muutnud võrgu võimekust märgatavalt. Igal sügavusel oli ülesobitamine tugev.

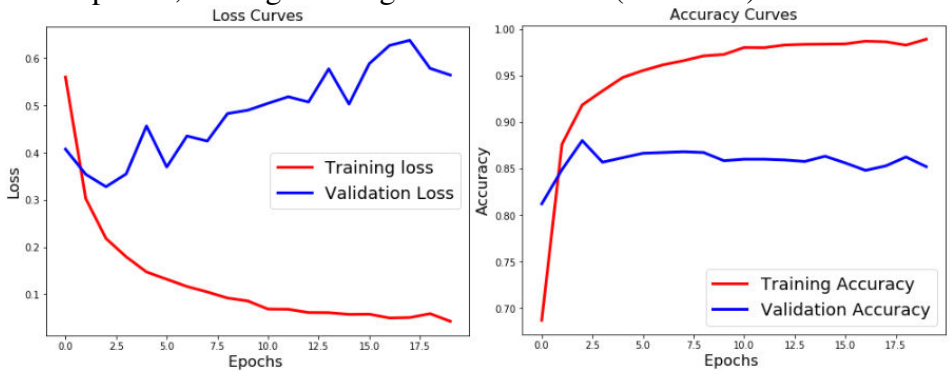


Joonis 36: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).



Joonis 37: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

Treenisin 45 plokiga 5 kihilist GRU võrku 20 epohhi, et saada ettekujutus sügavama GRU pikemast treenimisest. Parimaks validatsiooni tulemuseks oli 88%, kuid see saavutati 3-l epohhil, kus algas ka tugev ülesobitamine(Joonis 38).

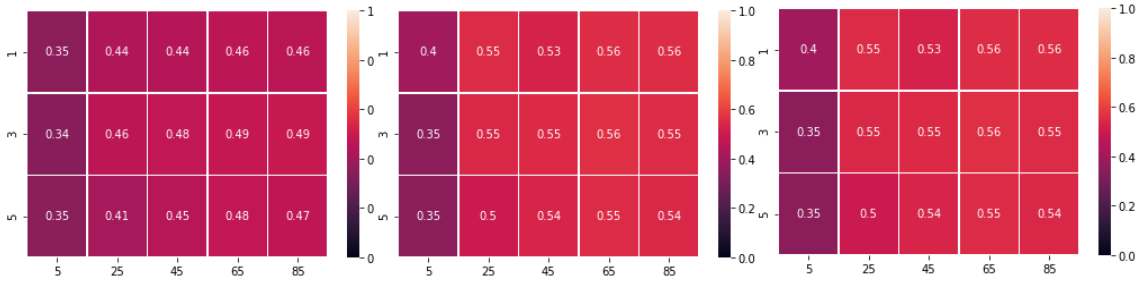


Joonis 38:45 plokiga 5 kihilise võrgu 20 epohhi treeningu tulemus.

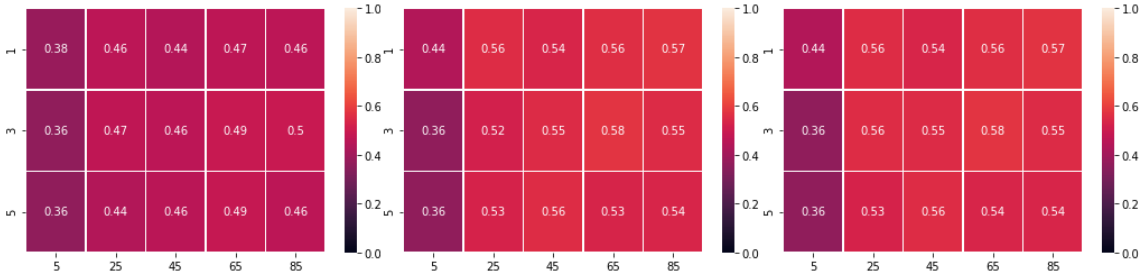
Tulemused olid väga sarnased LSTM närvivõrguga.

5.4.2 Reuters andmebaasil treenimine

Sarnaselt LSTM tulemutele olid GRU õhemad mudelid treenimisel aeglasemad (epohhide arvu poolest). Maksimaalne validatsiooni täpsus oli 58% ning maksimaalne treening täpsus oli 56%(Joonis 39-40). Kõik 5 ploki laiused kihid jäid 36% validatsiooni täpsusele kinni, 1 kihi sügavune mudel jõudis 10-ks epohhiks sealt välja. Validatsiooni tulemus 0.36197686 esines GRU mudelite treeningul 47 juhul. Kuna mõlema ploki tüübiga mudelites esines identne validatsiooni tulemus võib väita, et mõlemad mudelid õppisid klassifitseerima identseid artikleid ning, et andmebaas on ajaseeria treeningu jaoks liiga väike.

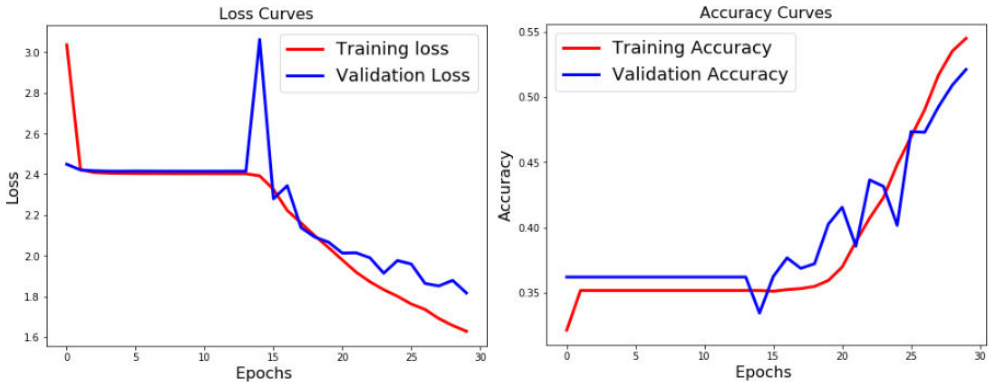


Joonis 39: Treening andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).



Joonis 40: Validatsiooni andmetel treenimise täpsuse heatmapid (keskmine tulemus vasakul, viimase epohhi tulemus keskel ja parim tulemus paremal).

Viie kihi sügavuse ning 5 plokki laiuse mudelit treeniti 30 epohhi, nägemaks kas mudel on võimeline 36% validatsiooni tulemuselt lahkuma. Mudel lahkus 36% platoolt 15-l epohhil. Maksimaalsed treening ja validatsiooni täpsused olid 54% ja 52% vastavalt, graafiku(Joonis 41) trend viitas potentsiaalsele tulemuse paranemisele. Lisa 30 epohhi treenimmine kasvatas parima tulemuse validatsiooni andmetel 53%-ni, treening täpsus kasvas üle 65%. 85 ploki laiuse ning 1 ja 3 kihi sügavustee mudelite 30 epohhi treeningu suurimad treening ja validatsiooni täpsused olid 70% ja 61%.



Joonis 41: 5x5 GRU 30 epohhi treenimise tulemus.

Sarnaselt LSTM-le ei mõjutanud sügavuse lisamine võrgu võimekust. Validatsiooni ja treening täpsused olid väga lähestikused ning validatsiooni ning treening erinevused olid väga sarnased.
Nullise täidise andmete lõppu tõstmisel ei suutnud 85 plokine 1 kihiline GRU 60 epohhiliselt treenimisel validatsiooni ega treeningu täpsus 36.95% ületada. Samade mõõtmetega LSTM ületas selle lävendi 45-dal epohhil jõudes 60-ks epohhiks 46% täpsuseni. GRU lisa 30 epohhiline treenimine tõstis validatsiooni täpsuse 40%-ni ja 60 epohhine 45%-ni.

6 Kokkuvõte

Bakalaureusetöö eesmargiks oli võrrelda erinevaid süvaõppe algoritme ja arhitektuure. Nende võrdluseks rakendati neid erinevatel benchmark andmebaasidel kasutades Kerase teeki. Põhiliseks võrdluse meetodiks oli erinevate mõõtmetega mudelite treenimine ning treeningu protsessist heatmapide genereeriminse tulemuste analüüs ning analüüsist juhenduv edasine uurimine.

Pildi klassifitseerimise andmebaasidel treenimisel selgus, et pärilevivõrgud on MNIST käsitsi kirjutatud numbrite klassifitseerimisel võrreldaval tugevusel konvolutsiooniliste närvivõrkudega. Mõlema arhitektuuri puhul piisas 98% validatsiooni täpsuse ületamiseks väga väikestest võrkudest. CIFAR10 ja CIFAR100 andmebaasidel olid konvolutsiooniliste närvivõrkude valideerimise tulemused pärilevi võrkudest tunduvalt suuremad, kuid esines väga palju ülesobitamist. Ülesobitamist parandas nii pärilevivõrkudel kui konvolutsioonilistel võrkudel dropouti lisamine. Lisaks dropouti lisamisele parandas konvolutsiooniliste võrkude klassifitseerimist ka ahenduskihtide lisamine. Ahenduskihtide lisamisel illustreeriti ka kaasnev treenitavate parameetrite arvu vähenemine. Pärilevivõrkudel demonstreeriti CIFAR andmebaasidel võrgu sügavuse väikene mõju klassifitseerimise tulemusele, põhiliseks klassifikatsiooni tulemust parandavaks faktoriks oli võrgu laiendamine. Konvolutsioonilistes

Rekurrentsetel mudelitel olid erinevused minimaalsed, treeningu ja validatsiooni tulemused olid mõne protsendi kauguses teineteisest. Ainus erinevus GRU ja LSTM jõudluses mida suudeti illustreerida oli andmete ettevalmistusel andmetäidise andmete lõppu panemine aeglustab GRU treenimist võrreldes LSTM-ga, mis demonstreeris LSTM sisendi avatuse kontrolli efektiivsust.

Töö käigus omandas autor kogemust pythoni programmeerimise keele, kerase teegi kasutamise ning Jupyter notebooki töökeskkonnaga. Töö käigus sai autor esimese kokkupuute masinõppe algoritmide programmeerimisega.

Kasutatud kirjandus

[1] S. Ruder, „An overview of gradient descent optimization,“ 2016.

[2] G. H. A. K. I. S. R. S. Nitish Srivastava, „Dropout: A Simple Way to Prevent Neural Networks from,“ *Journal of Machine Learning Research,* nr 15, pp. 1929-1958, 2014.

[3] A. M. S. B. Dominik Scherer, „Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition,“ *20th International Conference on Artificial Neural Networks,* September 2010.

[4] V. K. J. P. Daniel Svozil, „Introduction to multi-layer feed-forward neural networks,“ *Chemometrics and Intelligent Laboratory Systems ,* nr 39, pp. 43-62, 1997.

[5] R. C. C. G. M. S. Jean-Fran¸cois Couchot, „Steganalysis via a Convolutional Neural Network using Large Convolution Filters for Embedding Process with Same Stego Key,“ *European Research in Telemedicine,* kd. 6, nr 2, pp. 79-92, 2017.

[6] F. A. G. a. J. S. a. F. A. Cummins, „Learning to Forget: Continual Prediction with LSTM,“ *Neural Computation,* kd. 12, pp. 2451-2471, 2000.

[7] J. C. a. Ç. G. a. K. C. a. Y. Bengio, „Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,“ *CoRR,* kd. abs/1412.3555, 2014.

Lisa 1 – Töö jaoks kasutatud kood

https://github.com/suvaline/DeepLearningComparison