TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Arvutiteaduse instituut
Informaatika õppekava

Oliver Savolainen

Konvolutsiooniliste neurovõrkude kalibreerimine järkjärgulise külmutamise meetodiga

Bakalaureusetöö (9 EAP)

Juhendaja: Meelis Kull, PhD

Tartu 2023

Konvolutsiooniliste neurovõrkude kalibreerimine järkjärgulise külmutamise meetodiga

Lühikokkuvõte: Neurovõrke on väga edukalt kasutatud mitmete valdkondade ja ülesannete juures, neist üks parimaid näiteid on konvolutsiooniliste neurovõrkude kasutamine piltide klassifitseerimiseks. Samas ei saa tihti neid mudeleid siiski usaldada, sest nad on üldjuhul liiga enesekindlad oma ennustustes ehk määravad liiga kõrge tõenäosuse ennustatavale klassile.

Selle tõttu on vaja mudeleid kalibreerida. Probleemi lahendamiseks on loodud mitmeid kalibreerimismeetodeid, neist üks tõhusamaid on temperatuuri skaleerimine. Selles töös on kalibreerimiseks, täpsemalt liigse enesekindluse vähendamiseks uurimise all järkjärguline külmutamine. Järkjärgulise külmutamise all mõeldakse meetodit neurovõrgu treenimiseks, kus mingitel hetkel lõpetatakse valitud kihtide kaalude muutmine.

Töös kasutati Kängsepa 2018. aasta magistritöö lähtekoodi ja tulemusi. Kõigepealt leiti ühe mudeli ja andmestiku põhjal parimad viisid meetodi rakendamiseks. Valiti kaks järkjärgulise külmutamise skeemi ning seejärel implementeeriti Kängsepa töö abil mitmeid konvolutsioonilisi neurovõrke, rakendati neile järkjärgulist külmutamist ning treeniti neid. Saadud mudelite tulemusi võrreldi Kängsepa töö tulemustega. Kuigi ei saa väita, et uuritud mõõdikute põhjal aitas järkjärguline külmutamine liigset enesekindlust vähendada, siis vähendas külmutamine ajaliste ressursside kasutamist ning saavutas samas paljude mudelite juures sarnaseid tulemusi.

Võtmesõnad: konvolutsioonilised neurovõrgud, kalibreerimine, piltide klassifitseerimine, residuaalne neurovõrk, järkjärguline külmutamine

CERCS: P176 Tehisintellekt

Calibration of Convolutional Neural Networks with Gradual Freezing

Abstract: Neural networks have been successfully used for various tasks in various fields, with one of the best examples being the use of convolutional neural networks for image classification. However, these models often cannot be trusted, as they are frequently excessively confident in their predictions, assigning too high of a probability to the predicted class.

As a result, it is necessary to calibrate the models. Several calibration methods have been developed to address this problem, one of the most efficient ones is temperature scaling. This study investigates gradual freezing for calibration, specifically for reducing excessive confidence. Gradual freezing refers to a method of training in which the updating of weights for selected layers is stopped at certain moments.

The source code and results of Kängsepp's 2018 master's thesis were used in this work. First, the best ways to implement the method were found based on one model and dataset. Two gradual freezing schemes were chosen, and then several convolutional neural networks were implemented using Kängsepp's work and trained with gradual

freezing. The obtained models' results were compared with Kängsepp's results. Although it cannot be claimed that gradual freezing helped reduce excessive confidence based on the metrics used, freezing did reduce the time required for training while still achieving similar results for many models.

Keywords: convolutional neural networks, calibration, image classification, residual networks, gradual freezing

CERCS: P176 Artificial intelligence

Sisukord

Sissejuhatus

Sügavõpe on juba aastaid olnud väga tähtis osa paljude ülesannete ja valdkondade juures. Osade ülesannete juures, mis kasutavad sügavõpet, on jõutud inimestega sarnase sooituseni, heaks näiteks on piltide klassifitseerimine. Kasutades neurovõrke, täpsemalt konvolutsioonilisi neurovõrke, on mitmetel suurtel pildiandmestikel jõutud väga kõrge täpsuseni. [1]. Kuid tihti ei saa neid tulemusi usaldada. Kui mudel ennustab mingite piltide põhjal, mis klassi see pilt kuulub, siis tihti ta väljastab ennustades liiga suure tõenäosuse ühe klassi kohta ning liiga madalad tõenäosused teiste klasside kohta [2]. Sellistes olukordades võib tulemuste tõlgendaja arvata, et ennustus on kindlasti õige, aga tihti on ennustatud tõenäosus kõrgem kui reaalselt sellise tõenäosusega saadud täpsus.

Seda nimetatakse liigseks enesekindluseks, mis on konvolutsiooniliste neurovõrkude puhul tihti probleemiks. Selle parandamiseks kasutatakse kalibreerimist. Loodud on mitmeid kalibreerimismeetodeid, neist üks tõhusamaid on temperatuuri skaleerimine [2]. Selles töös on kalibreerimiseks, täpsemalt liigse enesekindluse vähendamiseks uurimise all järkjärguline külmutamine. Kihi külmutamine tähendab, et selle kihi kaale enam tagasilevi algoritmis ei muudeta, need jäävad samaks kuni treeningprotsessi lõpuni või kuni need pole enam külmutatud. Järkjärgulisel külmutamisel külmutatakse treeningprotsessi käigus mingitel hetkedel valitud kihte. Selle kasutamine aitab mudeli keerukust vähendada, mis võiks viia ülesobitamise ning liigse enesekindluse vähendamiseni.

Siinse töö eesmärk on implementeerida järkjärguline külmutamine ning otsida vastust küsimusele, kas see aitab vähendada konvolutsiooniliste neurovõrkude liigset enesekindlust. Samuti tahetakse leida, millised on parimad viisid selle meetodi rakendamiseks. Töös on kasutatud Markus Kängsepa 2018. aasta magistritöö lähtekoodi ja tulemusi [3]. Kui esimeses sisulises peatükis selgitatakse mõisteid ning järkjärgulise külmutamise kasutamist kalibreerimiseks, siis teises peatükis kirjeldatakse meetodi ja Kängsepa töös kasutatud mudelite implementeerimist ja esimesi eksperimente. Kolmandas peatükis on võrreldud lõplikke saadud tulemusi Kängsepa töö omadega.

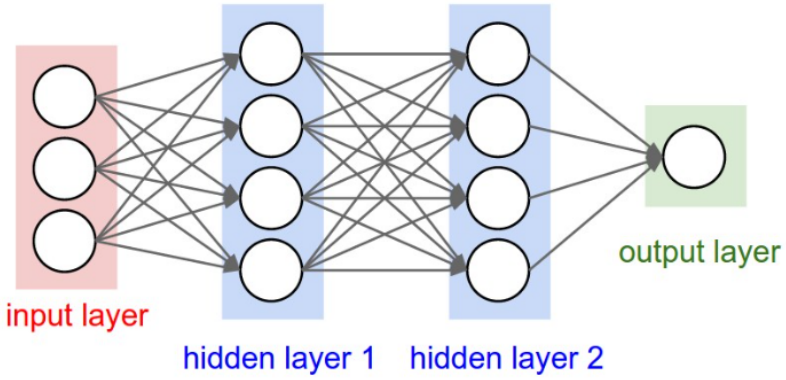
1 Mõisted ja definitsioonid

Selles töö osas selgitatakse peamiseid töös kasutatavaid mõisteid, definitsioone ning üldist teoreetilist tausta. Selle toel seletatakse peatüki lõpuks, miks kasutatakse järkjärgulist külmutamist liigse enesekindluse vähendamiseks.

1.1 Neurovõrk

Töös kasutatakse eesmärgi täitmiseks konvolutsioonilisi neurovõrke. Järgnevad kaks alampeatükki nende tutvustuseks on kirjutatud Goodfellowi jt [1] raamatu „Deep Learning“ põhjal.

Tehisnärvivõrk ehk neurovõrk (*neural network*) on arvutuslik arhitektuur, mis loodi ajust inspireerituna. Selle eesmärk on üldjuhul sisendi põhjal tagastada väljund millegi ennustamiseks. Tuntuimad neurovõrgud on pärilevi neurovõrgud (*feed-forward networks*), milles kõik neuronid on ühendatud suunaga väljundi poole moodustades suunatud tsükliteta graafi. Need koosnevad kihtidest, mis omakorda koosnevad omavahel ühendatud neuronitest ehk sõlmedest (*neurons*). Joonisel 1 on näha lihtne ülevaade pärilevi neurovõrgu arhitektuurist - väga tihti on neurovõrgul olemas sisendkiht (*input layer*) ja väljundkiht (*output layer*), üldjuhul ka nende vahel varjatud kihid (*hidden layers*). Igal kihil on sisendid, väljundid ja aktivatsioonifunktsioon (*activation function*), mis üldjuhul piirab neuroni väljundi vajalikku vahemikku. Neuronitevahelistel ühendustel on kaalud, millest sõltub sisendi tähtsus. Just kaalude väärtust muudetakse mudeli treeningprotsessi käigus.



Joonis 1. Neurovõrgu arhitektuuri ülevaade.

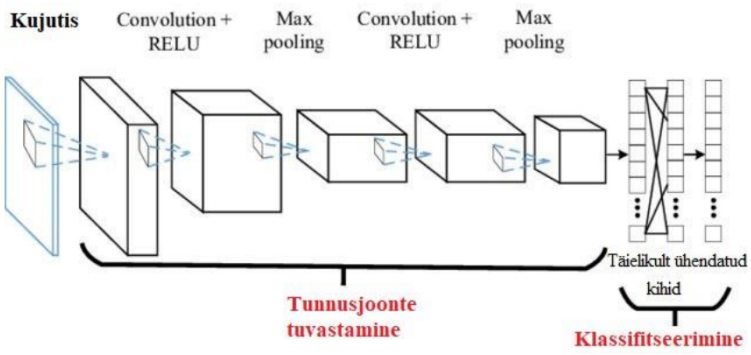
Nendes võrkudes rakendatakse tagasilevi algoritmi (*back-propagation*). Tagasilevi algoritmis leitakse kaofunktsiooni (*loss function*) väärtus ehk kui suur viga tehti ennustades ning vastavalt sellele muudetakse kaalusid alustades viimasest kihist kuni esimeseni. Eesmärk on minimeerida kaofunktsiooni väärtus ehk jõuda koondumiseni, mida tehakse muutuste suunda ja suurust näitava vektori abil, mida kutsutakse gradiendiks. Kui need muutused tehakse kõikide treeningandmete põhjal, on tegemist gradientlaskumisega. Kui aga ennustatakse väiksema osa ehk ploki põhjal gradiendi väärtust, siis on tegemist stohhastilise gradientlaskumisega (*stochastic gradient descent*).

See on kõige tavapärasem neurovõrkude optimeerimise algoritm, mida kasutatakse ka siin töös. Algoritmi töö sõltub paljuski hüperparameetrite väärtustest. Hüperparameetrid on väärtused, mida mudelid ei õpi treenides, need on enne kaasa antud. Gradiendi ennustused toimuvad iga miniploki (*batch*) järel, mille suurus üldjuhul on määratud hüperparameetriga. Kui käia läbi kõik treeningandmed miniplokkide kaupa, on läbitud üks epohh (*epoch*). Epohhide arv on üldjuhul hüperparameeter, mille järgi määratakse treeningu pikkus. Stohhastilist gradientlaskumist kasutades saab ette anda veel hüperparameetreid, näiteks õpisammu (*learning rate*). Õpisamm määrab, kui suur samm astutakse gradiendi suunas kaalusid muutes. Lisaks on siin töös kasutuses inertsitegur (*momentum*), mis akumuleerib eelnevate gradientide väärtuseid, et leida nende abil, kui palju tuleks kaalusid muuta, et kiiremini minimeerida kaofunktsiooni. Täpsemalt on siin töös tegemist Nesterovi inertsiteguriga (*Nesterov momentum*), kus akumuleerimine toimub pärast inertsiteguri rakendamist. Lisaks kõigile nendele meetoditele ja hüperparameetritele on treeningu kiirendamiseks kasutuses ka tipptasemel riistvara, eelkõige GPUd ehk graafikaprotsessorid (*Graphics processing units*) - graafikakaartidel asuvad mikroprotsessorid, mis suudavad paralleelselt palju arvutusi sooritada ning tänu sellele sobivad väga hästi neurovõrkude treenimiseks.

Kasutades kirjeldatud arhitektuuri ja meetodeid, saab neurovõrgule ette anda lihtsalt kõik andmepunktid ning tihti jõuda väga hea tulemuseni. Võrreldes mitmete teiste masinõppe mudelitega pole vaja ise tunnuseid eraldi valida või luua. Piisab lihtsalt andmepunktidest, millel on vastav väljund ette antud ja mudel leiab ise, milliseid tunnuseid tasub kasutada. Veidi erinevat arhitektuuri kasutatakse aga siis, kui on palju sisendeid, kuid sisendid võivad vaid osade väljunditega seotud olla. Kõige tavalisemalt on sisendiks 2D pildi andmed, kus kolmes järjendis on iga pildi piksli kohta vastavalt sinise, punase või rohelise värvi intensiivsuse väärtus. Sellisel juhul kasutatakse enamasti konvolutsioonilisi neurovõrke.

1.2 Konvolutsiooniline neurovõrk

Konvolutsioonilistel neurovõrkudel on erinevalt tavalistest pärilevi neurovõrkudest lisaks konvolutsioonilised kihid, mis koosnevad kolmest etapist. Joonisel 2 on toodud abstraktne joonis, kuidas tunnuseid nende abil leitakse, selles sisalduvaid mõisteid kirjeldatakse järgmistes lõikudes.



Joonis 2. Konvolutsioonilise neurovõrgu tööd selgitav abstraktne joonis [4].

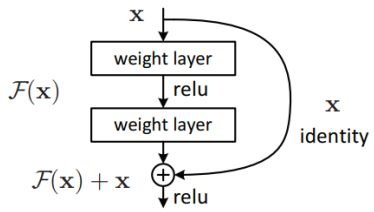
Esimeses etapis toimuvad matemaatilised operatsioonid nimega konvolutsioonid (*convolutions*), kus kahe funktsiooni põhjal luuakse kolmas funktsioon, mis kirjeldab, kuidas üks funktsioon mõjutab teist. Sisendiks olevad kaks funktsiooni konvolutsioonilistes neurovõrkudes on mitmedimensionaalne järjend andmeid ja mitmedimensionaalne järjend õpitavaid parameetreid, väljundiks olev funktsioon kannab üldjuhul tunnuste kaardi (*feature map*) nimetust.

Teises etapis rakendatakse mittenegatiivset lineaarfunktsiooni ehk ReLU, mis asendab tunnuste kaardi negatiivsed väärtused nullidega. See toob mittelineaarsust võrku ehk aitab leida andmetest keerulisemaid tunnuseid.

Kolmandas etapis tuleb kasutusse ahendus (*pooling*), mis asendab üksikute neuronite väljundid nende naabrite kokkuvõtva väljundiga. See vähendab väikeste muutuste tähtsust. Konvolutsioonikihte ja kirjeldatud etappe esineb kõikides konvolutsioonilistes neurovõrkudes, aga mudelid võivad muude osade tõttu siiski selgelt erineda.

Selles töös on kasutuses neli konvolutsiooniliste neurovõrkude arhitektuuri: ResNet, ResNet SD, Wide ResNet ja DenseNet. Kõik need on tuntud mudelid, mida kasutati Kängsepa 2018. aasta magistritöös [3].

ResNet ehk residuaalne neurovõrk (Residual Network) on konvolutsiooniline neurovõrk, mis kasutab lisaks tavalisele konvolutsioonilise neurovõrgu arhitektuurile jääkühendusi ehk annab varasemate kihtide sisendi otse hilisematele kihtidele samasusteisendusega (*identity function*) [5]. Näide sellest on toodud joonisel 3.



Joonis 3. Residuaalse neurovõrgu jääkühenduste kasutus [5].

ResNet SD ehk residuaalne neurovõrk stohhastilise sügavusega (*ResNet with Stochastic Depth*) on ResNeti modifikatsioon, mis kasutab stohhastilist sügavust ehk lülitab juhuslikult välja osad kihid treeningprotsessis, mis suurendab kiirust ja vähendab ülesobitamist [6] . Wide ResNet on ResNeti variant, kus kihid on laiemad, aga neid on vähem ehk mudeli sügavus on väiksem [7]. DenseNet (*Dense Convolutional Network*) omab erinevalt tavalistest konvolutsioonilistest neurovõrkudest $L(L+1)/2$ ühendust iga kihi ja sellele järgneva kihi vahel, L tähistab mitmenda kihiga on tegemist [8]. Neid kõiki mudeleid kasutatakse tõenäosuslike klassifikaatoritena.

1.3 Tõenäosuslik klassifikaator

Flachi 2012. aasta ülevaate kohaselt [9] on klassifikaator funktsioon, mis andmepunkti X tunnuste väärtuste põhjal tagastab hulka Y kuuluva klassimärgendi. Binaarne klassifikaator ennustab vaid kahe erineva kahe klassi korral. Üldjuhul toimub see sündmuse toimumise kohta. Mitmeklassiline klassifikaator ennustab seda rohkem kui kahe klassi korral. Näiteks binaarne klassifikaator ennustab vaid, kas sademeid tuleb või mitte, mitmeklassiline võib aga ennustada, kas tuleb vihma, lund, rahet või ei tule sademeid.

Tõenäosuslik klassifikaator on funktsioon, mis iga $x \in X$ korral tagastab tõenäosusvektori $(p_1, \ldots, p_k)$, kus p_i tähistab ennustatud tõenäosust, et nende väärtuste põhjal vastab x-le i-nda klassi märgend [9]. Tõenäosusvektor tagastatakse neurovõrkude korral üldjuhul softmax funktsiooniga, mis viib mudeli arvutatud reaalarvulised väljundid vahemikku 0-st 1-ni, kus kõigi väärtuste summa on kokku täpselt 1 [2]. Töös kasutataksegi klassifikaatori mõistet tõenäosusliku klassifikaatori tähenduses ja proovitakse just selle määramatust kalibreerida.

1.4 Määramatuse kalibreerimine

Olgu kasutuses binaarne klassifikaator, kus ennustatakse sisendi põhjal, kas märgend on 0 või 1. Enesekindlus olgu siin ennustatud tõenäosus, et märgend on 1. Määramatust kalibreerides tahetakse vähendada enesekindluse ja reaalse tõenäosuse vahelist eksimust.

Reaalne tõenäosus olgu siin märgendiga 1 andmepunktide osakaal nendest andmepunktidest, mille korral klassifikaator on vaadeldava enesekindlusega. Näiteks tahetakse, et andmepunktid, mille korral on klassifikaator 70% enesekindlusega, oleksid 70% korral juhtudest märgendiga 1. Kui see on täpselt nii ehk eksimus on 0, on mudel perfektselt kalibreeritud. Samas ei tohi kalibreerides täpsus ideaalis üldse kehvemaks minna, sest täpsus on klassifitseerides ikkagi kõige tähtsam mõõdik. Mitmeklassilise klassifikaatori korral aga nii lihtsalt kalibreeritust hinnata ei saa, sest kalibreerida on vaja üldjuhul mitme klassi ennustusi [2].

Kull jt [10] on defineerinud klassifikaatori kalibreerimise kolmel erineval viisil. Mitmeklassiliselt kalibreeritud või lihtsalt kalibreeritud on klassifikaator $\hat{p}$ siis, kui iga tõenäosusvektori $q = (q_1, \ldots, q_k)$ korral vastab väljundiks olev vektor $\hat{p}(x) = (\hat{p}_1(x), \ldots, \hat{p}_k(x))$ tegelikule klassijaotusele. Valemis (1) on kirjas, kuidas see väljendub:

$$P(Y = i \mid \hat{p}(X) = q) = q_i \text{ iga } i = 1, \ldots, k \text{ korral.} \tag{1}$$

Enesekindluse järgi kalibreerituks nimetatakse klassifikaatorit, kui kõikide ennustavate maksimaalsete tõenäosuste c korral vastab tegelikult sellesse klassi kuuluvate andmepunktide hulk c väärtusele. Seda väljendab valem (2):

$$P\left(Y = \arg\max\left(\hat{p}(X)\right) \mid \max\left(\hat{p}(X)\right) = c\right) = c. \tag{2}$$

Klassifikaatorit nimetatakse klassikaupa kalibreerituks, kui iga klassi i korral vastab iga ennustatav tõenäosus $\hat{p}_i(X)$ tegelikult sellesse klassi kuuluvate andmepunktide hulgale. Välja on toodud see valemis (3):

$$P(Y = i \mid \hat{p}_i(X) = q_i) = q_i. \tag{3}$$

Kui klassifikaator on mitmeklassiliselt kalibreeritud, siis on ta enesekindluse ja klassikaupa kalibreeritud, vastupidiselt ei pruugi see kehtida [10].

Olgu kasutuses klassifikaator, mis ennustab, kas tuleb vihma, lund, rahet või ei tule sademeid. Kogu andmestikus suurusega 10 olgu 4 andmepunkti, mille märgend vastab vihmale, 2 lumele vastava märgendiga andmepunkti, 1 rahele vastava märgendiga andmepunkt ning ülejäänud 3 puhul olgu tegemist sademeteta klassi kuuluvate andmepunktidega. Kui klassifikaatori ennustuseks iga andmepunkti korral on tõenäosusvektor (0,4; 0,2; 0,1; 0,3), on see kõikide definitsioonide järgi kalibreeritud. Mitmeklassiliselt on see mudel kalibreeritud, sest selle väljund vastab tegelikult klassijaotusele (0,4;0,2;0,1;0,3). Enesekindluse järgi on kalibreeritud, sest maksimaalne tõenäosus 0,4 väljundis vastab enim esineva klassi tõenäosusele. Klassikaupa on mudel kalibreeritud, sest 0,4 vastab vihma tähistava klassi kuuluvatele andmepunktide osakaalule, 0,2 lund tähistava klassi kuuluvatele andmepunktide osakaalule, 0,1 rahe klassi andmepunktide osakaalule ja 0,3 sademeteta klassi andmepunktide osakaalule. Definitsioonidele vastamine aga ei tähenda, et praktiliselt on käes suurepärane mudel, sest selle täpsus oleks selle näite

korral vaid 40%, sest iga kord ennustatakse, et tegemist on vihmaga. See näide näitab, et definitsioonidest ainult ei piisa, et leida, kui hästi kalibreeritud klassifikaator on, vaja on teistsuguseid mõõdikuid.

1.5 Kalibreerimise mõõtmine ja meetodid

Selle töö aluseks on võetud Kängsepa 2018. aasta magistritöö [3], mis omakorda toetub Guo 2017. aasta tööle [11]. Selles alampeatükis on kirjeldatud Kängsepa töös valitud kalibreerimise mõõdikuid ning meetodeid, mida käsitletakse ka selles töös.

Naeni jt [12] kohaselt on eeldatav kalibreerimisviga ehk ECE mõõdik, mis hindab mudeli kalibreerimise täpsust, näidates prognoositud tõenäosuste ja tegelike sageduste vahelist erinevust leides kaalutud keskmise täpsuse ja enesekindluse vahel M tulba kohta. Täpsemalt toimub see n ennustuse korral valem (4) järgi:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|. \tag{4}$$

Valemis (5) on toodud suurima sellise vahe leidmine, mis on ühtlasi ka suurima kalibreerimisvea ehk MCE leidmine:

$$MCE = \max_{m \in \{1,\dots,M\}} |acc(B_m) - conf(B_m)|. \tag{5}$$

.

Neurovõrkude mõõtmisel kasutatakse tihti negatiivset log-tõepära ehk NLLi (*Negative Log Likelihood*). Täpsemalt hinnatakse mudeli ebakindlust leides prognoositud tõenäosuste ja tegelike tulemuste vahelised erinevused. Olgu y vektor, kus y_i on väärtusega 1 vaid siis, kui klassi i korral on tegemist tõese klassimärgendiga, vastasel juhul on selle väärtus 0. Sel juhul saab NLLi leidmise panna kirja valemiga (6) [2]:

$$NLL = -\sum_{i=1}^{k} y_i \log(\hat{p}_i(X)) \tag{6}$$

Kull jt 2019. aasta artikli kohaselt [10] on mudelite puhul probleem eelkõige liigse enesekindlusega, mis tähendab, et väljundiks olevas tõenäosusjaotuses on ennustatavate klasside tõenäosused liiga kõrged. Liigne enesekindlus tuleneb üldjuhul treeningandmestikule ülesobitamisest. Enamus kalibreerimismeetodid proovivad seda probleemi lahendada pärast tavalise treeningprotsessi lõppu. Üks levinumaid viise selleks on Guo jt loodud [11] temperatuuri skaleerimine. Temperatuur on reaalarvuline väärtus, mis õpitakse treeningandmestikust eraldi oleva andmestiku peal, minimeerides mõõdikuna NLLi. Seda kasutatakse uute enesekindluste leidmiseks valemiga (7), kus T on temperatuur, z_i on viimaselt kihilt saadud reaalarvuliste väärtuste vektor ja σ_{SM} softmax funktsioon:

$$\hat{c}_i = \max_k \sigma_{SM}\left(\frac{z_i}{T}\right)(k) \tag{7}$$

Temperatuuriga jagatakse läbi mudeli viimaselt kihilt saadud reaalarvulised väärtused enne kui need tõenäosusteks teisendatakse. See ei muuda mudeli täpsust, aga peaaegu garanteeritult parandab kalibreeritust.

Kängsepa töös on käsitletud veel teisigi meetodeid. Beeta kalibreerimisel kasutatakse NLLi mõõdikuna, et leida kolme parameetri väärtused paremaks kalibreerimiseks. Histogrammimeetod ja isotooniline regressioon kasutavad andmepunktide tulpadeks jagamist ja minimeerivad kindla kaofunktsiooni väärtust. Need kolm meetodit jäid aga Kängsepa töös enamuse mõõdikute osas selgelt alla temperatuuri skaleerimisele, kui võrreldi väärtusi samadel mudelitel ja andmestikel, mida ka selles töös kasutatakse. Vaid täpsuse, täpsemalt veamäära ehk ebakorrektsete ennustuse osa juures olid beeta kalibreerimise tulemused paremad. Selles töös võrreldakse järkjärgulise külmutamise tulemusi nende nelja kirjeldatud meetodiga, rõhuasetus on võrdlusel temperatuuri skaleerimise tulemustega.

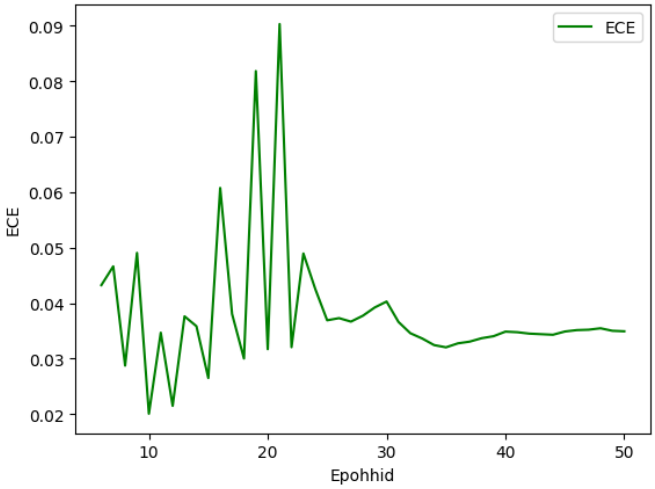
1.6 Järkjärguline külmutamine

Kihi külmutamine tähendab, et selle kihi kaale enam tagasilevi algoritmis ei muudeta, need jäävad samaks kuni treeningprotsessi lõpuni või kuni need pole enam külmutatud. Järkjärgulisel külmutamisel külmutatakse treeningprotsessi käigus mingitel hetkedel valitud kihte. Kuigi seda ja sarnaseid mõisteid on erinevates töödes kasutatud ka muu järjekorraga külmutamisel, siis siin töös mõistetakse seda kui külmutamist, mis hakkab ühel hetkel esimesest kihist ja toimub järjestikku kuni viimase kihini, mis külmutatakse. Kõik külmutatud kihtide kaalud jäävad muutumatuks kuni treeningprotsessi lõpuni. Kirjeldust kõikidest külmutamistest, sealhulgas nende külmutamise hetkedest ja külmutatavate kihtide arvust, nimetatakse siin töös järkjärgulise külmutamise skeemiks.

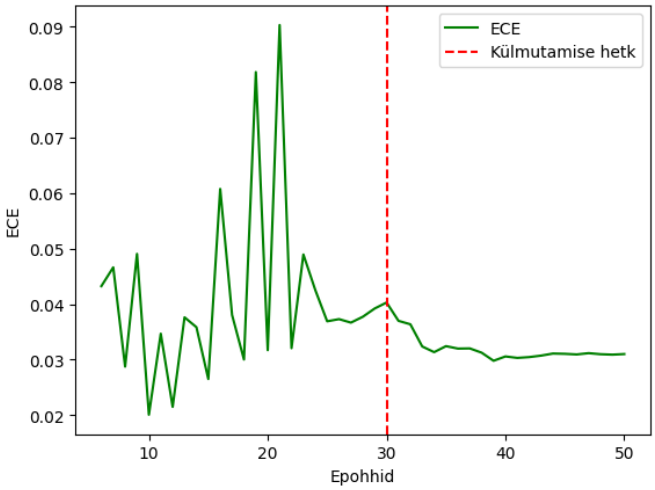
Üldjuhul on seda meetodit kasutatud treeningu kiirendamiseks. Seda on kasutanud näiteks Xiao jt [13] oma uurimuses, kus erinevalt sellest tööst arvutati gradiendi muutumise järgi, millal ning millised kihid külmutada. Uurimuses selgus, et selle tehnikaga on võimalik saavutada populaaarse pildiandmestiku CIFAR-10 klassifitseerimisel peaaegu sama täpsus kui ilma külmutamata. Küll aga läks treening selgelt kiiremaks. Romero jt [14] kasutasid siirdeõppe raames tehnikat viisil, kus alustasid hilisemate kihtide treenimisega ja hiljem treeningprotsessis tõid kaasa esimeste kihtide treenimise. Seda tegid nad eeldusega, et varasemate kihtide kaalud väljendavad rohkem üldiseid tunnuseid ja hilisematega saab paremini peenhäälestada mudelit kindla andmestiku jaoks. Seda eeldust saab nende töö põhjal lugeda enam-vähem tõeseks, sest selle meetodiga suutsid mudelid olla sama täpsed kui külmutamata mudeli tulemustega.

Romero jt töös kasutatud eelduse põhjal võiks ka arvata, et järkjärguline külmutamine saab olla efektiivne kalibreerimismeetod. Kui varasemate kihtide optimaalsemate kaaludeni on võimalik jõuda varem kui hilisemate kihtide juures, siis võib nende muutmine kogu treeningprotsessi käigus olla ebavajalik ja isegi viia liigse enesekindluseni. Näide sellest on toodud joonistel 4 ja 5. Joonistel on näidatud ECE muutumine testandmestikul

50 epohhi kestvas treeningus. Joonis 4 on külmutamiseta treening, joonisel 5 on tulemused külmutamise skeemi korral, kus 30. epohhi juures külmutatakse 80% kihtidest. Täpsemalt on kirjeldatud nende jooniste saamiseks kasutatud mudelit, treeningut ja skeemide leidmist järgmises peatükis.



Joonis 4. Külmutamata treenitud konvolutsioonilise neurovõrgu ECE muutumine.



Joonis 5. Konvolutsioonilise neurovõrgu ECE muutumine pärast 30. epohhi juures külmutamist.

Jooniste võrdluses on näha, et ECE väärtus tõuseb umbes viimase 20 epohhi juures külmutamata treenides. Külmutamise järel aga trend on pigem ühtlane ning ECE väärtus on lõpuks madalam kui 30. ja 35. epohhi juures. See on näide sellest, et järkjärguline külmutamine võiks liigse enesekindluse vastu aidata. Kuna külmutades on tagasilevi algoritmis vähem kaalusid, mida muudetakse, läheb mudel vähem keerukaks ning võib vähem ülesobituda treeningandmestikule, mis ühtlasi võiks ka aidata vähendada liigset enesekindlust. Võrdluse saab ka tuua temperatuuri skaleerimisega, kus sisuliselt külmutatakse kõik peale viimase kihi ning treenitakse seda. Selge erinevus on küll see, et see viimane kiht on üks parameeter ning selle väärtus leitakse teise andmestiku peal, üldjuhul valideerimisandmestikul. Varem pole järkjärgulist külmutamist kalibreerimise eesmärgiga uuritud, järgnevas peatükis kirjeldataksegi, kuidas seda esmakordselt tehakse.

2 Järkjärgulise külmutamise meetodi implementeerimine ja uurimine

Töö eesmärk on leida, kas järkjärgulise külmutamise meetod aitab vähendada konvolutsiooniliste neurovõrkude liigset enesekindlust tuntud pildiandmestikel ning milline on parim viis selle meetodi rakendamiseks. Selleks kirjeldatakse järgnevalt, kuidas järkjärgulise külmutamise meetod implementeeriti ja eksperimendid selle uurimiseks läbi viidi.

2.1 Ülevaade järkjärgulise külmutamise implementeerimisest

Selles alampeatükis on välja toodud järkjärgulise külmutamise implementeerimiseks kasutatud tarkvara, kasutatud andmestikud ning implementeerimise protsess. Eksperimentide kood on kirjas repositooriumis, mille link on toodud lisas 1.

2.1.1 Kasutatud tarkvara

Töös on kasutatud Markus Kängsepa 2018. aasta magistritöö "Calibration of Convolutional Networks" [3] lähtekoodi, sest selles on varasemate artiklite põhjal implementeeritud tuntud konvolutsiooniliste neurovõrkude arhitektuurid, nende treeningprotsess ja lisaks rakendatud ka tuntud kalibreerimismeetodeid. See teeb üsna lihtsaks kõikide nende mudelite loomise ja treenimise ning annab võimaluse võrrelda, kuidas järkjärguline külmutamine tulemusi mõjutab. Kogu lähtekood on kirjutatud Pythoni programmeerimiskeeles ning mudelite loomiseks ja treenimiseks on kasutatud Kerase raamistikku [15]. Lisaks on kasutatud NumPy, pandase ja scikit-learni raamistikke erinevate arvutuste ja masinõppe protseduuride tegemiseks.

Samuti on kasutatud ChatGPT abi nii järkjärgulise külmutamise implementeerimisel kui ka mudelite treenimisel. Üldjuhul anti mudelile sisendiks veateade ning vastuse abil prooviti vigu lahendada või tahetava abimeetodi kirjeldus ning vastuseks saadud koodi abil loodi vastavaid meetodeid. Lisaks on ChatGPTd kasutatud valemite, tabelite ning viidete vormistamiseks.

2.1.2 Kasutatud andmestikud

Nagu eelnevalt kirjeldatud, implementeeriti Kängsepa töös [3] konvolutsioonilisi neurovõrke mõnedel tuntud andmestikel ja rakendati nendele mudelitele mitmeid kalibreerimismeetodeid [3]. Võimalikult heaks võrdluseks nende tulemustega on kasutatud ka siin samu arhitektuure ja andmestikke, mille treeningprotsessi on vaid lisatud järkjärguline külmutamine.

Eelnevas peatükis välja toodud mudelitega leiti täpsus ja kalibreerimistulemused neljal erineval pildiandmestikul. Nendeks on CIFAR-10, CIFAR-100 ja SVHN nimelised andmestikud ning lisaks esimeses eksperimentide etapis FMNISTi andmestik.

CIFAR-10 koosneb 60000 pildist, mis on võrdselt jagatud 10 erinevasse klassi: lennuk, auto, lind, kass, põder, koer, konn, hobune, laev ja veoauto [16]. CIFAR-100 andmestik on sarnane, aga 100 erineva klassiga [16]. SVHN (*Street View House Numbers*) on kogumik värvilisi pilte majanumbritest, kus on vaja klassifitseerida üks keskne number, kuigi numbreid võib pildil veel olla [17]. Kui need andmestikud koosnevad kõik värvilistest piltidest 32x32 dimensioonidega, siis FMNIST (*Fashion-MNIST*) sisaldab must-valgeid 28x28 pilte, mis jagunevad kümnesse klassi riideesemete järgi [18]. Need asjaolud teevad selle andmestiku peal treenimise kiiremaks kui teistel andmestikel, mis oli vajalik esimese etapi eksperimentide tegemiseks.

Andmestikud jaotati treening-, valideerimis- ning testandmestikeks Kängsepa töö kirjelduste järgi [3]. CIFAR-10 ja CIFAR-100 korral oli treeninguks 45000 pilti, valideerimiseks 5000 pilti ning testimiseks 10000 pilti. SVHN andmestikul treenides oli 598388 pilti treeningandmestikus, 6000 pilti valideerimisandmestikus ning 26032 testandmestiku pilti. Lisaks oli FMNISTi kasutades treeninguks 6000 pilti, valideerimiseks 5400 pilti ning testimiseks 10000 pilti.

Andmestike juures tuleb mainida ka andmete eeltöötlust ning rikastamist. SVHN ja FMNIST andmetele tehakse eeltöötlust lahutades iga piksli kolme põhivärvi väärtustest ehk kanalitest selle kanali treeningandmestiku keskmine väärtus ning jagatakse see tulemus samamoodi leitud standardhälvega [6]. Sarnaselt tehakse CIFAR-10 ja CIFAR-100 andmestikel, ainult ResNeti mudeli korral leitakse keskmine ja standardhälve kõikide kanalite peale kokku [11]. Selline eeltöötlus viib kõik väärtused vahemikku, mis sobib konvolutsioonilisele neurovõrgule paremini [1]. Lisaks kasutatakse CIFAR-10 ning CIFAR-100 andmestike juures andmete rikastamist, kus pööratakse osasid pilte ning lisatakse igasse äärde juurde 4 pikslit, mille järel kärbitakse juhuslikult pilte algsesse suurusesse tagasi [11]. See lisab treeningandmestikku rohkem andmeid, mis võiks mudelit treenimisel paremaks teha [1].

2.1.3 Järkjärgulise külmutamise implementeerimine

Kängsepa repositooriumis oli juba olemas mudelite arhitektuur ning nende treenimis- ja hindamisprotsess, sealhulgas kalibreerituse hindamine varem mainitud nelja mõõdikuga [3]. Lisada tuli aga järkjärgulise külmutamise osa treenimisele. Külmutamine oli alati alates esimesest kihist järjest kuni viimase külmutatavani. See tähendas, et treenimisele on ette antud iga mudeli kihi jaoks epohhi arv, mille juures seda enam ei treenita ehk selle kaalud enam ei muutu. Pärast ühe või mitme kihi külmutamist tuleb mudel uuesti kompileerida ja alles siis saab seda edasi treenida kuni järgmise epohhini, kus järgmiste kihtide külmutamine toimub.

Mudelitel oli Kängsepa lähtekoodis iga andmestiku juures määratud õpisammu

väärtuse muutumine epohhi arvust olenevalt. Tavaliselt saab need muutused ette anda Kerase vaikemeetodiga, mis loeb epohhe alates viimasest mudeli kompileerumisest. Kuna aga järkjärgulise külmutamisega oli vaja kompileerida mudelit korduvalt, oli vaja luua uus meetod, mis loeks kõik epohhid kokku ja muudaks õpisammu sellest olenevalt. Võrreldes Kängsepa lähtekoodiga oli vaja ka muuta ja lahendada üksikuid probleeme, mis olid tekkinud nii Kerase kui ka teiste raamistike muutustest pärast algse koodi kirjutamist. Näiteks oli vaja muuta osade meetodite nimesid mudelite loomisel või muuta andmestruktuure, mida ette anda NLLI arvutamiseks. Pärast probleemide lahendamist sai alustada eksperimenteerimisega.

2.2 Järkjärgulise külmutamise eksperimendid

Kui järkjärguline külmutamine oli implementeeritud ning kood korduvalt testitud erinevate mudelite ja andmestike puhul, sai alustada protsessi eksperimentide läbi viimiseks.

Algne plaan oli lokaalsete või muude lihtsalt kättesaadavate ressurssidega teha algseid teste, mida kutsutakse siin töös esimeseks etapiks. Teises etapis oli plaan viia edasi 1 kuni 3 parimat järkjärgulise külmutamise skeemi täisandmestike peal testimiseks. Algupäraste eksperimentide tulemused pole aga välja toodud. Need eksperimendid koosnesid tihti mitmetest erinevatest külmutamisest, mis tähendas, et ei saanud selgeks teha, kuidas mõjutasid erinevad muutused treeningprotsessis tulemust. Edasiselt muudeti veidi lähenemist, selle kirjeldus on toodud järgmises alampeatükis.

2.2.1 Eksperimenteerimise esimese etapi kirjeldus

Esimeses etapis sooritati teste lihtsama ja väiksema andmestiku peal arvutusressurside kokkuhoidmiseks, et leida parimad variandid külmutamiseks, mida hiljem rakendada Kängsepa töös kirjeldatud treeningutele.

Selles etapis tehtud eksperimendid on tehtud FMNISTi andmestikul ResNet SD mudeliga, millel on 152 konvolutsioonikihti. Vaja oli leida andmestik, mida oleks võimalik ühe mudeliga treenida koondumiseni 5 kuni 10 minutiga, et kõik eksperimendid oleks võimalik mõistliku aja sees ära teha. Kängsepa töös kasutatutest oli ehk kiireimaks treeninguks parim SVHN andmestik, sest see sisaldas vaid numbreid [3], kuid ka vaid 5% treeningandmeid kasutades ei olnud soovitud ajavahemik piisav treenimiseks. Seega tuli proovida veel lihtsamat andmestikku, aga ideaalis vältida nii lihtsat andmestikku nagu näiteks vaid käsikirjas numbreid sisaldav MNIST, sest selle peal on võimalik väga häid tulemusi saada üsna lihtsate mudelitega. Valikusse tuli FMNISTi andmestik, mille peal ei saada nii häid täpsuse tulemusi nagu MNISTi peal, aga tegemist on siiski mustvalgete 28x28 dimensioonidega piltidega ehk treenimine on kiirem kui kõikidel teiste mainitud andmestike korral, kus kasutuses värvilised 32x32 pildid. Kahjuks ka selle korral sai kasutada vaid 10% treeningandmetest, et 10 minutiga koondumise lähedale jõuda treenides. Kuna kogu see proovimine algas SVHN andmestiku peal, mida treeniti

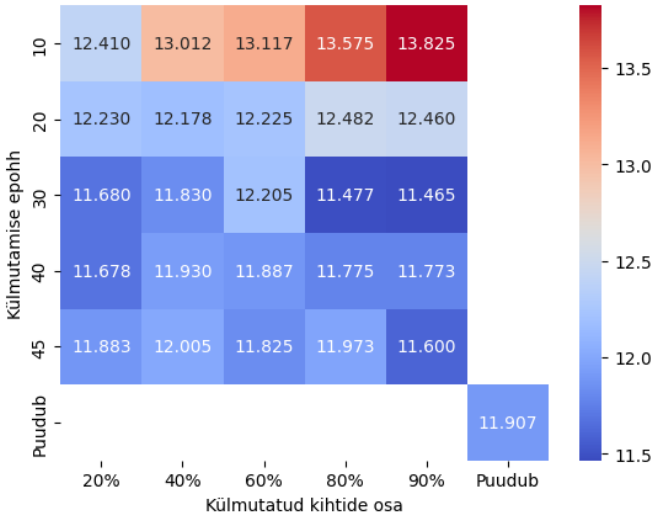
ResNet SD mudeliga, siis sai ka see mudel valitud nendeks eksperimentideks. See on küll otsus, mida võiks muuta tagasi vaadates, sest kuigi oli soov kasutada mudelit, mida kasutatakse ka teises etapis, siis oleks võinud ehk valida väiksema mudeli, mis oleks jõudnud soovitud ajavahemikus koonduda suurema andmestiku korral.

Treening kestis 50 epohhi ning selle käigus muutus õpisamm sarnase tempoga nagu paljude Kängsepa töös tehtud treeningute korral [3], kus õpisammu väärtus langes 10 korda 50% epohhide juures ja veel 10 korda, kui 75% epohhidest oli möödunud. Täpsemalt oli õpisammu väärtus siin esimesed 25 epohhi 0,01, järgmised 15 epohhi 0,001 ning viimased 10 epohhi 0,0001. Tuleb küll märkida, et kõik need väärtused on 10 korda madalamad kui tavaliste täisandmestikega treeningute korral, sest vastasel juhul ei saanud mudel koonduda.

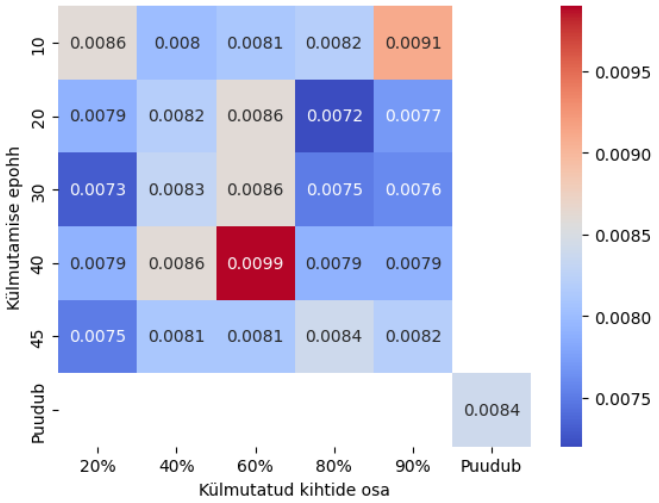
Esimeses eksperimentide etapis valiti 5 eri epohhide arvu, millal külmutada, et võrrelda külmutamist olenevalt epohhide arvust ning külmutatud kihtide arvust. Epohhide arvude valikuteks olid 10, 20, 30, 40 ja 45, see andis võimaluse võrrelda külmutamise hetki kõigi erinevate õpisammu väärtuste juures. Nende epohhide juures külmutati 20%, 40%, 60%, 80% või 90% kihtidest. Igat kombinatsiooni jooksutati 8 korda, et saada iga kombinatsiooni kohta võimalikult usaldatav tulemus. Usaldatavuse suurendamiseks arvestati neist 4 keskmisega ehk eemaldati on 2 parimat ja 2 halvimat tulemust, et juhuslikult väga head või väga kehvad tulemused eemaldada. Lisaks on kõikidele mudelitele rakendatud temperatuuri skaleerimist, sest see oli kalibreerimise jaoks Kängsepa töös parim meetod ning ka siin töös parandas ECE ja NLLi tulemusi [3]. Eelkõige oli eesmärk nii esimeses kui teises eksperimentide etapis võrrelda, kas järkjärguline külmutamine vähendab liigset enesekindlust määral, mida on osade mõõdikute arvestuses näha pärast kalibreerimismeetodi rakendamist. Esimese etapi tulemused on toodud järgmises alampeatükis.

2.2.2 Eksperimenteerimise esimese etapi tulemuste analüüs

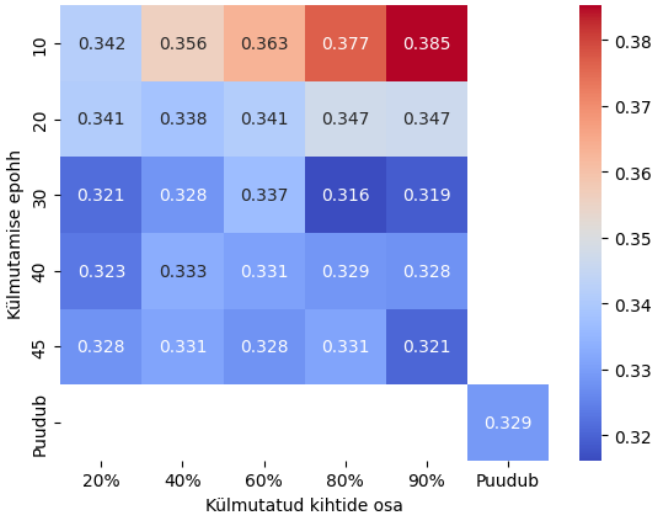
Kombinatsioonide võrdlust veamäära, ECE ja NLLi arvestuses on näha vastavalt joonistel 6, 7 ja 8. Tulemused on leitud pärast temperatuuri skaleerimist. Kõikide näitajate puhul kehtib reegel, et madalam tulemus on parem. Seega on nendel soojuskaartidel (*heatmaps*) tumesinised lahtrid parimate tulemustega ning tumepunased halvimate tulemustega. "Puudub"rea ja veeru nimega ruut tähistab tulemust, kus külmutamist ei rakendatud.



Joonis 6. Soojuskaart kombinatsioonide veamäära võrdlusest.



Joonis 7. Soojuskaart kombinatsioonide ECE võrdlusest.



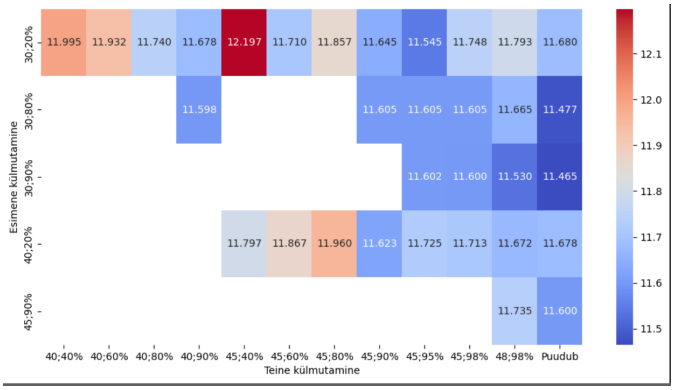
Joonis 8. Soojuskaart kombinatsioonide NLLi võrdlusest.

Pärast temperatuuri skaleerimist olid ECE tulemused väga sarnased. Seega ei olnud mõtet selle mõõdiku järgi siin etapis otsuseid teha. Pigem tuli lähtuda veamäära ja NLLi väärtustest, eriti võrreldes ilma külmutamata saadud tulemusega, mida võib siin kutsuda etaloniks. Nendes arvestustes jäid kõik enne 30. epohhi tehtud külmutamised alla etalonile. Samas enamus hiljem tehtud külmutamistest suutis parandada algset tulemust mõlema mõõdiku osas. Mõõdikutest enam peaks arvestama NLLi, sest antud töö on

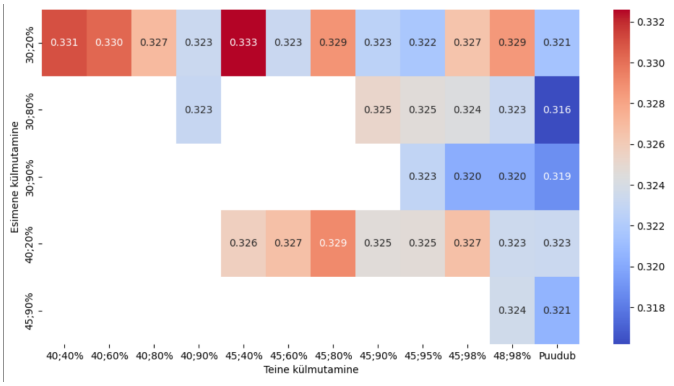
kalibreerimise kohta.

Just viis parimat kombinatsiooni NLLi väärtuse järgi sai valitud järgmisesse alametappi, kus nendele külmutamistele lisati teine külmutamine. Need olid 30. epohhi juures 20% kihtide külmutamine, 30. epohhi juures 80% kihtide külmutamine, 30. juures 90% külmutamine, 40. juures 20% külmutamine ja 45. juures 90% külmutamine. Teise külmutamisena tehti läbi kõik variandid algsetest kombinatsioonidest, mis olid selle esimese külmutamise korral võimalikud ning lisati ka osad hilisemad ja veel rohkemate kihtidega külmutamised, mis olid 45. juures 95% kihtide külmutamine, 45. juures 98% külmutamine, 48. juures 95% külmutamine ja 48. juures 98% külmutamine.

Kui esimesed külmutamised näitasid, et vähemalt selle mudeli ja andmestiku korral on mitmeid kombinatsioone, mis edestavad tavalist treeningut kõikide mõõdikute osas, siis teiste külmutamistega ei leidunud ühtegi kombinatsiooni, mis oleks toonud täpsuse ja NLLi korral edasisi paranemisi nagu joonistel 9 ja 10 on näha. Neist küll 30. juures 90% külmutamine ja hilisemad külmutamised tõid kaasa väga sarnased tulemused esimestele külmutamistele, aga üldjuhul teised külmutamised tõid kaasa veamäära ja NLLi suurenemise.



Joonis 9. Soojuskaart kahe külmutamise kombinatsioonide veamäära võrdlusest.



Joonis 10. Soojuskaart kahe külmutamise kombinatsioonide NLLi võrdlusest.

Seega tuli vastu võtta otsus kolmandaid külmutamisi enam mitte teha ning liikuda edasi teise etappi parimate skeemidega. Valitud sai kaks. Esiteks parima NLLi saavutanud 30. epohhi juures 80% kihtide külmutamine, mis sai tähistuse S1. Teiseks 30. epohhi juures 90% külmutamise ja 48. epohhi juures 98% külmutamise kombinatsioon, sest see saavutas ainult 0,001 kehvema NLLi kui lihtsalt 30. epohhi juures 90% külmutamine, aga erineb esimesest valitud skeemist rohkem ning läheb kokku paremini töö esimeses peatükis mainitud temperatuuri skaleerimise võrdlusega. Teine skeem sai tähistuse S2. Järgnevas alampeatükis on kirjeldatud läbi viidud treeninguid, kus enamus treeningutel on epohhide arv erinev kui 50. Seega on esimeses etapis leitud külmutamise hetked võetud suhtarvuna kogu epohhide arvust ehk 30. epohhi puhul 60% epohhide arvust ja 48. epohhi puhul 96% epohhide arvust.

2.2.3 Eksperimenteerimise teise etapi kirjeldus

Teises etapis treenitakse kõik varem mainitud mudelid vastavalt Kängsepa töö kirjeldustele ja lähtekoodile rakendades juurde kahte valitud järkjärgulise külmutamise skeemi [3]. Selles alampeatükis on kirjeldatud, kuidas need treeningud läbi viidi.

Ressurssidena kasutati siin Tartu Ülikooli High Performance Computing Centre ressursse [19], täpsemalt Nvidia Tesla V100 GPUd ja Kaggle ressursse [20], täpsemalt Nvidia Tesla P100 GPUd. Esimest kasutades esines tõrkeid ja ajalimiite, mis nõudsid meetodi loomist, mis suudaks eelmist treeningut jätkata kohast, kus viimane kaalude fail oli salvestatud.

Kõik treeningud tehti Kängsepa töö [3] lähtekoodi ja kirjelduste järgi. Järgnevates lõikudes on välja toodud ka viited varasematele artiklitele, mille järgi Kängsepa töösse hüperparameetrite väärtused ja implementatsioon valiti. Optimiseerimise algoritm oli iga treeningu juures stohhastiline gradientlaskumine, kus Nesterovi inertsitegur oli väärtusega 0,9. Miniploki suurus oli 128, ainus erand oli DenseNeti treenimine. Kõi-

kide treeningute juures leiti tulemused pärast kogu epohhide arvu ja lisaks testiti ka pärast igat epohhi saadud kaalusid valideerimisandmestikul ning lisati parim mudel valideerimisandmestikul tulemuste võrdlusesse.

ResNeti mudel oli 110 konvolutsioonikihti sisaldav ning seda treeniti CIFAR-10 ja CIFAR-100 andmestikel kasutades Li 2017. aasta artikli lähtekoodi [21] ning He jt 2015. aasta artikli kirjeldusi [5]. Treening kestis kokku 200 epohhi ehk esimese skeemi esimene külmutamine toimus pärast 120. epohhi. Õpisamm oli esimesed 80 epohhi 0,1, järgmised 70 epohhi 0,01 ning lõpus 0,001.

ResNet SD mudelit treeniti Huangi jt 2016. aasta artikli põhjal [6], implementatsioon pärineb Cheni 2016. aasta koodist [22]. CIFAR-10 andmestikul treenides ei hakanud see mudel alguses koonduma. Võimalus oleks olnud alustada väiksema õpisammuga, kuid see oleks võrdluse saamiseks nõudnud ka ilma külmutamata treenimist ja testimist, milline õpisamm toimiks. Seega toimus treening vaid CIFAR-100 ja SVHN andmestikel. CIFAR-100 andmestiku mudel oli 110-konvolutsioonikihiline ning treening kestis kokku 500 epohhi, õpisamm oli väärtusega 0,1 250 epohhi, 0,01 kuni 375. epohhini ning viimased 125 epohhi 0,001. SVHN andmestikul treenimine oli kokku vaid 50 epohhi, 30. epohhi juures langes õpisamm 0,1 pealt 0,01 peale ning 35. epohhi juures läks see veel 10 korda väiksemaks. Lisaks oli esimesed 2 epohhi õpisamm 0,04, et koondumine toimuks.

Wide ResNet oli siin 32 konvolutsioonikihiga neurovõrk, mille implementatsioon pärines Majumdari 2018. aasta lähtekoodist [23]. Hüperparameetrite väärtused treeninguks tulid Zagoruyko and Komodakise 2016. aasta artiklist [7] . Treening kestis CIFAR-10 ja CIFAR-100 andmestikel 200 epohhi. Õpisamm algas 0,1 peal ning langes 20 korda nii 60., 120. ja 160. epohhi juures.

DenseNeti mudel koosnes 40 konvolutsioonikihist, selle implementatsioon on võetud Majumdari 2018. aasta [24] ning Yu 2017. aasta repositooriumitest [21]. Treeningud toimusid CIFAR-10 ja CIFAR-100 andmestikel Huangi jt 2018. aasta töös kirjeldatud hüperparameetrite väärtuste põhjal [8] . Need olid ainsad treeningud, kus miniploki suurus oli 64. Õpisamm oli 0,1 esimesed 150 epohhi, 0,01 järgmised 75 ning 0,001 viimased 75 epohhi.

3 Eksperimenteerimise teise etapi tulemuste analüüs

Siin peatükis analüüsitakse teise etapi tulemusi veamäära, NLLi ja ECE osas. Kasutuses on sarnased tabelid nagu Kängsepa töös ning järkjärgulise külmutamise skeemide tulemusi on võrreldud Kängsepa töö tulemustega [3]. Nende võrdluste tabelites on toodud mudeli nimi, andmestik, ilma külmutamata tulemus ehk Algne ning seejärel viie kalibreerimismeetodi tulemused. Histo tähistab histogrammimeetodit, Iso isotoonilist regressiooni, Temp temperatuuri skaleerimist ning Beeta beeta kalibreerimist.

Nende järel on 4 veergu selle töö tulemusi. S1 tähistab esimese skeemi ehk 60% epohhide juures 80% kihtide külmutamise lõplike kaalude tulemust ehk mudelit, mis on saadud pärast kõikide epohhide möödumist. S2 tähistab teise skeemi ehk 60% epohhide juures 90% kihtide ning 96% epohhide juures 98% kihtide külmutamise lõplike kaalude tulemust. S1* tähistab parimat leitud mudelit valideerimisandmestiku põhjal esimese skeemiga treenides ning S2* parimat leitud mudelit valideerimisandmestiku põhjal teise skeemi korral. Kõik tulemused on leitud pärast temperatuuri skaleerimist, sest see oli Kängsepa töös kõige edukam kalibreerimismeetod enamus mõõdikute arvestuses [3]. Kalibreerimata tulemuste võrdlused NLLi ja ECE arvestuses on toodud lisades 2 ja 3. Iga tulemuse kohta on toodud selle järk vaadeldava treeningu kohta alaindeksina ning lõpuks leitud keskmine järk iga meetodi kohta. Lisaks nende kolme mõõdiku võrdlustele on peatükis toodud ka järkjärgulise külmutamise ajalise mõju analüüs.

3.1 Veamäära võrdlus

Tabelis 1 on toodud veamäära võrdlus. See on mõõdik, mille väärtust temperatuuri skaleerimine ei muuda. See oli ka mõõdik, kus beeta kalibreerimine suutis temperatuuri skaleerimist edestada, aga teised meetodid muutsid täpsuse kalibreerimata mudelitest kehvemaks [3].

Tabel 1. Veamäära (%) võrdlus mudelite ja andmestike kaupa.

Mudel	Andmestik	Algne	Histo	Iso	Temp	Beeta	S1	S2	S1*	S2*
DenseNet 40	CIFAR-10	$7{,}58_4$	$7{,}93_8$	$7{,}65_7$	$7{,}58_4$	$7{,}59_6$	$7{,}57_3$	$7{,}99_9$	$7{,}49_2$	$\mathbf{7{,}37}_1$
DenseNet 40	CIFAR-100	$30{,}0_3$	$32{,}49_7$	$30{,}22_6$	$30{,}0_3$	$\mathbf{29{,}81}_1$	$30{,}02_5$	$29{,}87_2$	$41{,}0_9$	$36{,}88_8$
ResNet 110	CIFAR-10	$6{,}44_2$	$6{,}59_5$	$\mathbf{6{,}36}_1$	$6{,}44_2$	$6{,}44_2$	$7{,}15_8$	$6{,}93_6$	$7{,}17_9$	$7{,}0_7$
ResNet 110	CIFAR-100	$28{,}52_2$	$31{,}26_9$	$29{,}31_4$	$28{,}52_2$	$\mathbf{28{,}36}_1$	$29{,}98_6$	$30{,}32_8$	$29{,}91_5$	$30{,}31_7$
ResNet SD 110	CIFAR-100	$27{,}17_6$	$29{,}74_9$	$27{,}47_8$	$27{,}17_6$	$26{,}32_5$	$\mathbf{24{,}89}_1$	$25{,}54_3$	$25{,}4_2$	$26{,}28_4$
ResNet SD 152	SVHN	$1{,}85_2$	$2{,}07_5$	$1{,}96_4$	$1{,}85_2$	$\mathbf{1{,}82}_1$	$2{,}3_6$	$2{,}77_8$	$2{,}38_7$	$2{,}77_8$
Wide ResNet 32	CIFAR-10	$6{,}07_5$	$6{,}18_7$	$5{,}98_3$	$6{,}07_5$	$5{,}94_2$	$\mathbf{5{,}93}_1$	$6{,}04_4$	$9{,}09_9$	$6{,}51_8$
Wide ResNet 32	CIFAR-100	$26{,}18_4$	$28{,}89_7$	$26{,}33_6$	$26{,}18_4$	$25{,}98_3$	$\mathbf{25{,}72}_1$	$25{,}83_2$	$32{,}8_9$	$32{,}48_8$
Keskmine järk		$3{,}5_2$	$7{,}12_9$	$4{,}88_5$	$3{,}5_2$	$\mathbf{2{,}62}_1$	$3{,}88_4$	$5{,}25_6$	$6{,}5_8$	$6{,}38_7$

Mõlema skeemi lõplikute kaaludega mudelid on üsna selgelt paremad kui valideerimisandmestiku parimad mudelid, kuigi Kängsepa töös leiti nii DenseNeti kui ResNet SD

tulemused just valideerimisandmestiku parimate mudelite järgi. Väljaspool paari erandit oli S1* ja S2* tulemused ühed viimastest kõikide treeningute korral. Esines ka väga kehvasid tulemusi nende korral, näiteks DenseNet 40 mudelit CIFAR-100 andmestikul treenides olid S1* ja S2* tulemused külmutamata variandist vastavalt üle 10% ja 6% kehvemad.

S1 ja S2 keskmise leitud järgu põhjal temperatuuri skaleerimise ning beeta kalibreerimise tulemust ei edestanud. S1 suutis küll mõne treeningu juures kõiki teisi meetodeid edestada, kuid näiteks ResNet 110 mudeli osas oli üks kehvemaid. Sarnast kõikumist esines ka S2 korral, kuid üldkokkuvõttes oli see teine skeem esimesest veidi kehvem. Erinevused polnud aga kunagi väga suured ning jäid pea kõikide võrdluste osas alla 2%, tihti olid need veel väiksemad. Ühe erandina ResNet SD 110 mudeli CIFAR-100 treenimisel suutis esimene skeem kõiki Kängsepa töös leitud tulemusi üle 2% edestada.

3.2 NLLi võrdlus

NLL oli esimeses etapis mõõdik, mille järgi lõplikud otsused tehti. Selle tähtsus ei vähene ka selles etapis ehk selle tulemusi saab tõsiselt arvestada järelduste tegemisel.. Kängsepa töös tulid parimad tulemused üsna selgelt temperatuuri skaleerimisega, ainult beeta kalibreerimine jõudis lähedale temperatuuri skaleerimise tulemustele [3]. Tabelis 2 on näha võrdlused selle töö tulemustega.

Tabel 2. NLLi võrdlus mudelite ja andmestike kaupa.

Mudel	Andmestik	Algne	Histo	Iso	Temp	Beeta	S1	S2	S1*	S2*
DenseNet 40	CIFAR-10	$0{,}4282_8$	$0{,}5725_9$	$0{,}2773_7$	$0{,}2251_3$	$0{,}2392_6$	$0{,}2249_2$	$0{,}2391_5$	$0{,}2259_4$	$\mathbf{0{,}2187}_1$
DenseNet 40	CIFAR-100	$2{,}0174_8$	$4{,}1829_9$	$1{,}6491_7$	$\mathbf{1{,}0571}_1$	$1{,}1532_4$	$1{,}0705_2$	$1{,}0705_2$	$1{,}4647_6$	$1{,}3186_5$
ResNet 110	CIFAR-10	$0{,}3583_8$	$0{,}5472_9$	$0{,}2708_7$	$0{,}2093_2$	$0{,}2139_4$	$0{,}226_6$	$0{,}2101_3$	$0{,}2249_5$	$\mathbf{0{,}2079}_1$
ResNet 110	CIFAR-100	$1{,}6937_7$	$4{,}2139_9$	$1{,}8926_8$	$1{,}0917_3$	$1{,}1318_4$	$1{,}1389_5$	$1{,}165_6$	$\mathbf{1{,}0621}_1$	$1{,}0794_2$
ResNet SD 110	CIFAR-100	$1{,}3525_7$	$4{,}0187_9$	$1{,}6371_8$	$0{,}9421_5$	$0{,}9643_6$	$\mathbf{0{,}8745}_1$	$0{,}8922_3$	$0{,}891_2$	$0{,}9108_4$
ResNet SD 152	SVHN	$0{,}0854_3$	$0{,}2887_9$	$0{,}1093_6$	$\mathbf{0{,}0786}_1$	$0{,}0796_2$	$0{,}0993_4$	$0{,}1124_7$	$0{,}1006_5$	$0{,}1126_8$
Wide ResNet 32	CIFAR-10	$0{,}3817_8$	$0{,}5132_9$	$0{,}2327_6$	$0{,}1915_3$	$0{,}202_5$	$0{,}1854_2$	$\mathbf{0{,}1822}_1$	$0{,}2653_7$	$0{,}192_4$
Wide ResNet 32	CIFAR-100	$1{,}8022_8$	$3{,}9352_9$	$1{,}5607_7$	$0{,}9445_2$	$1{,}0386_4$	$0{,}9466_3$	$\mathbf{0{,}9423}_1$	$1{,}1557_6$	$1{,}1311_5$
Keskmine järk		$7{,}12_8$	$9{,}0_9$	$7{,}0_7$	$\mathbf{2{,}5}_1$	$4{,}38_5$	$3{,}12_2$	$3{,}5_3$	$4{,}5_6$	$3{,}75_4$

Esimese eksperimentide etapi tulemused näitasid, et pärast temperatuuri skaleerimist on NLLi tulemused selgelt veamääraga korreleeruvad. S1* ja S2* keskmised järgud olid paremad kui veamäära osas, kuid jäid lõplike kaalude tulemustele üldjuhul alla.

NLLi arvestuses on S1 ja S2 tulemused paremad ilma külmutamata temperatuuri skaleerimisest täpselt nende treeningute osas, kus tulemused olid ka paremad veamäära osas. Sarnaselt eelmisele alampeatükile on vahed üsna väiksed kahe skeemi lõplike kaalude tulemuste ning Kängsepa töö temperatuuri skaleerimise tulemuste vahel. NLLi võrdluses on aga kõikidel teistel kalibreerimismeetoditel selgelt kehvemad tulemused, sest järkjärgulise külmutamise mudelitele on rakendatud temperatuuri skaleerimist, seega on S1 ja S2 keskmised järgud madalamad.

3.3 ECE võrdlus

ECE numbrid on tabelis 3 toodud protsendina. Selle arvestuses oli Kängsepa töös selgelt parim meetod temperatuuri skaleerimine. Teised meetodid jäid selgelt alla.

Tabel 3. ECE (%) võrdlus mudelite ja andmestike kaupa.

Mudel	Andmestik	Algne	Histo	Iso	Temp	Beeta	S1	S2	S1*	S2*
DenseNet 40	CIFAR-10	$5{,}5_{9}$	$2{,}13_{8}$	$1{,}68_{6}$	$0{,}95_{5}$	$1{,}7_{7}$	$0{,}93_{4}$	$\mathbf{0{,}63}_{1}$	$0{,}87_{3}$	$\mathbf{0{,}63}_{1}$
DenseNet 40	CIFAR-100	$21{,}16_{9}$	$11{,}97_{8}$	$5{,}25_{6}$	$\mathbf{0{,}9}_{1}$	$6{,}03_{7}$	$0{,}91_{2}$	$0{,}91_{2}$	$1{,}34_{5}$	$1{,}09_{4}$
ResNet 110	CIFAR-10	$4{,}75_{9}$	$1{,}25_{6}$	$1{,}47_{8}$	$1{,}13_{3}$	$1{,}42_{7}$	$1{,}21_{5}$	$0{,}75_{2}$	$1{,}19_{4}$	$\mathbf{0{,}61}_{1}$
ResNet 110	CIFAR-100	$18{,}48_{9}$	$9{,}06_{8}$	$6{,}54_{7}$	$2{,}38_{4}$	$4{,}6_{6}$	$1{,}77_{3}$	$2{,}55_{5}$	$\mathbf{1{,}62}_{1}$	$1{,}7_{2}$
ResNet SD 110	CIFAR-100	$15{,}86_{9}$	$7{,}6_{8}$	$5{,}21_{7}$	$1{,}21_{4}$	$3{,}55_{6}$	$1{,}15_{3}$	$\mathbf{0{,}96}_{1}$	$1{,}55_{5}$	$0{,}98_{2}$
ResNet SD 152	SVHN	$0{,}86_{9}$	$0{,}53_{3}$	$\mathbf{0{,}25}_{1}$	$0{,}61_{4}$	$0{,}5_{2}$	$0{,}81_{8}$	$0{,}69_{5}$	$0{,}76_{7}$	$0{,}7_{6}$
Wide ResNet 32	CIFAR-10	$4{,}51_{9}$	$1{,}01_{7}$	$1{,}19_{8}$	$0{,}8_{4}$	$0{,}97_{6}$	$0{,}58_{2}$	$\mathbf{0{,}52}_{1}$	$0{,}85_{5}$	$0{,}59_{3}$
Wide ResNet 32	CIFAR-100	$18{,}78_{9}$	$7{,}64_{8}$	$5{,}82_{6}$	$\mathbf{0{,}78}_{1}$	$7{,}01_{7}$	$1{,}6_{3}$	$1{,}98_{5}$	$1{,}11_{2}$	$1{,}77_{4}$
Keskmine järk		$9{,}0_{9}$	$7{,}0_{8}$	$6{,}12_{7}$	$3{,}25_{3}$	$6{,}0_{6}$	$3{,}75_{4}$	$\mathbf{2{,}75}_{1}$	$4{,}0_{5}$	$2{,}88_{2}$

ECE arvestustes suutis S2* saavutada parema keskmise järgu kui S1, kuid jällegi olid mõlema skeemi korral lõplike kaalude keskmised järgud paremad kui valideerimisandmestiku parima tulemuse järgi valides.

S2 oli ECE võrdluses enamus treeningute korral parem kui tavaline temperatuuri skaleerimine, eriti selgelt paranes seda skeemi kasutades kalibreeritus ECE arvestuses CIFAR-10 andmestikul. S1 tulemused olid vaid kahe treeningu puhul paremad kui S2 omad, aga olid külmutamata temperatuuri skaleerimisest neljal puhul madalama ECE väärtusega.

3.4 Ajaline võrdlus

Tabelites 4 ja 5 on toodud vastavalt esimese ja teise skeemi ajalised võrdlused enne ja pärast külmutamisi, kõik ajad on sekundites. Esimeses veerus on keskmine epohhi ajaline pikkus enne külmutamist ning teises veerus summaarne aeg selle põhjal. Järgnevates veergudes on keskmised epohhide pikkused pärast külmutamisi, esimese skeemi puhul oli külmutamisi üks ning teise skeemi korral kaks. Eelviimases veerus on tegelik summaarne aeg. Lõpuks on toodud kui palju kogu aeg vähenes protsentides. Siin tuleb küll mainida, et arvutus on tehtud puhtalt epohhide aja järgi, välja on jäetud protsessid enne treenimist, epohhide vahel toimuvad protsessid nagu valideerimisandmestikul ennustamine ja failide salvestamine ning arvutused pärast treeningut. Nende sisse arvestamisel suureneksid mõlemad koguajad üsna sarnasel määral ehk vähenemise protsendid ilmselt veidi langeksid. Kuna aga esines tõrkeid nagu näiteks internetiühenduse peatumine või koguni treeningu katkemine, siis tundus kõige mõistlikum võrrelda puhtalt kaua epohhid kokku aega võtsid.

Tabel 4. Esimese skeemi ajaline võrdlus sekundites

Mudel	Andmestik	Epohhi pikkus	Eeldatav koguaeg	Epohh pärast külmutamist	Tegelik koguaeg	Koguaja kiirenemine
DenseNet 40	CIFAR-10	40,17	12051	31,33	10990,2	8,8%
DenseNet 40	CIFAR-100	48,56	14568	29,04	12225,6	16,08%
ResNet 110	CIFAR-10	58,9	11780	38,85	10284	12,7%
ResNet 110	CIFAR-100	58,57	11714	38,56	10113,2	13,67%
ResNet SD 110	CIFAR-100	37,16	18580	31,56	17460	6,03%
ResNet SD 152	SVHN	454,65	22732,5	185,3	17345,5	23,7%
Wide ResNet 32	CIFAR-10	126,96	25392	55,58	19636,8	22,67%
Wide ResNet 32	CIFAR-100	126,96	25392	55,7	19691,2	22,45%

Tabel 5. Teise skeemi ajaline võrdlus sekundites

Mudel	Andmestik	Epohhi pikkus	Eeldatav koguaeg	Epohh pärast külmutamist	Epohh pärast teist külmutamist	Tegelik koguaeg	Koguaja kiirenemine
DenseNet 40	CIFAR-10	39,82	11946	30,38	29,68	10804,8	9,55%
DenseNet 40	CIFAR-100	48,45	14535	28,77	28,5	12170,16	16,27%
ResNet 110	CIFAR-10	44,02	8804	29,4	27,92	7622,56	13,42%
ResNet 110	CIFAR-100	35,9	7180	28,18	27,08	6553,6	8,72%
ResNet SD 110	CIFAR-100	44,13	22065	29,6	28,82	19143,4	13,24%
ResNet SD 152	SVHN	449,85	22492,5	163,15	147,03	16726,26	25,64%
Wide ResNet 32	CIFAR-10	126,51	25302	50,08	50,14	19154,88	24,29%
Wide ResNet 32	CIFAR-100	127,38	25476	49,78	45,99	19237,68	24,49%

Eeldada võis, et külmutamised kiirendavad treeningut, sest kaalusid muudetakse vähem. Tulemustest on näha, et nii on. Ainus kord, kui kiirenemine jäi alla 8%, oli ResNet SD 110 CIFAR-100 andmestikul treenimisel esimese skeemiga. Kõige enam aega võtnud mudelite osas jõudis kiirenemine mõlema skeemi korral üle 20%. Teine skeem külmutab rohkem kihte ja on seega kiirem enamus treeningute osas. Kõik vähenemised on piisavalt suured, et kiirendada selgelt treenimist, isegi kui arvestada sisse järkjälgulise külmutamise protsessid.

3.5 Järeldused

Võrreldes mudeleid, mida on treenitud järkjärgulise külmutamise kahe valitud skeemiga, täpsemalt võrreldes tulemusi pärast temperatuuri skaleerimise rakendamist, ei saa väita, et järkjärguline külmutamine kindlasti aitab vähendada konvolutsiooniliste neurovõrkude liigset enesekindlust. Teine skeem oli küll keskmise järgu võrdluses ECEd mõõtes veidi parem, kuid jäi täpsuses ja seetõttu ka NLLi arvestuses alla ilma külmutamata treenimisele. Esimene skeem sisaldas vähemate kihtide külmutamist sama epohhi juures ning seega üsna loogiliselt viis teisest skeemist parema täpsuseni, aga selle puhul ei saa väita, et kalibreeritus paranes. Kõige kindlamalt saab iga kasutatud mõõdiku juures väita, et järkjärgulise külmutamise meetodi osas on treeningu lõplikud kaalud paremad kui parimad leitud kaalud valideerimisandmestiku järgi.

See, et järkjärguline külmutamine selles töös ei toonud selgeid paranemisi nii täpsuse kui ka kalibreerimise osas, ei tähenda, et järkjärgulist külmutamist konvolutsiooniliste neurovõrkude osas ei võiks edasi uurida. Esiteks tuleb mainida, et selles töös olid piiratud ressursid ja aeg, et leida paremad skeemid. Kui uurida selgelt rohkem kombinatsioone, eelistatavalt rohkemate mudelite ja andmestike peal kui esimeses etapis tehtud eksperi-

mendid ühe andmestiku ja mudeliga, on ilmselt võimalik leida paremaid kombinatsioone kui siin töös leiti. Saaks ka teha paremaid valikuid mudeli osas, ResNet SD 152 polnud ilmselt parim valik andmestikku ja ressursse arvestades esimeses etapis. Saaks ka proovida teisi lähenemisi skeemide leidmiseks. Võiks kaaluda näiteks Xiao jt [13] meetodi kasutamist, mis suutis CIFAR-10 peal mitme erineva mudeliga saavutada pea sama või veidi parema täpsuse kui tavaline treenimine.

Teiseks on siin töös ikkagi leitud, et üsna sarnaseid tulemusi on võimalik saavutada nii kalibreerituse kui täpsuse osas võrreldes külmutamata variandiga vähendades samal ajal 6-25% treenimiseks kuluvat aega. See tähendab, et rakendades järkjärgulist külmutamist, saab sama aja ja ressursidega treenida suuremaid ja keerulisemaid mudeleid kui algselt plaanis. Kahe skeemi lõplike kaalude tulemuste peale leidus vaid üks kord, kus veamäär oli rohkem kui 1,5% suurem kui tavalise treeningu tulemus ja ka siis oli vahe 1,8%. NLLi tulemused olid veel lähedamal ning ECE pigem paranes teise skeemiga. Sellised muutused ei ole väga negatiivsed ning saab väita, et päris tihti on külmutamisega saadud aja säästmine väärtuslikum, eriti kui tänu sellele on võimalik treenida suurem ja parem mudel.

Kokkuvõte

Töös kirjeldati konvolutsiooniliste neurovõrkude tööd, kalibreerimist, järkjärgulist külmutamist ning põhjuseid selle kasutamiseks kalibreerimismeetodina. Eksperimente tehti kahes etapis. Esimeses leiti ühe mudeli ja andmestiku põhjal kaks parimat järkjärgulise külmutamise skeemi. Neid rakendati teises etapis mitmete mudelite treenigutele ning tulemusi võrreldi Kängsepa 2018. aasta töö tulemustega.

Eesmärgiks oli leida, kas järkjärgulise külmutamise meetod aitab konvolutsiooniliste neurovõrkude liigset enesekindlust vähendada tuntud pildiandmestikel ning milline on parim viis selle meetodi rakendamiseks. Esimeses etapis leiti kaks skeemi. Esimene ehk S1 sisaldas 60% epohhide juures 80% kihtide külmutamist. Teine ehk S2 koosnes 60% epohhide juures 90% kihtide külmutamise ja 96% epohhide juures 98% kihtide külmutamisest. Nendele rakendati veel temperatuuri skaleerimist. Kuigi S2 suutis kalibreerimismõõdiku ECE arvestuses edestada tavalist temperatuuri skaleerimist, oli see kehvem NLLi ja veamäära arvestuses. S1 oli viimase kahe mõõdiku arvestusest teisest skeemist parem, kuid jäi siiski üldjuhul tavalisele temperatuuri skaleerimisele alla. Samas olid mõlemad skeemid üsna selgelt kiiremad tavalisest treeningust.

Edasiarendusena saaks proovida veel erinevaid skeeme, võrreldes neid rohkem kui ühe mudeli puhul. Samuti näitavad tulemused, et järkjärgulist külmutamist rakendades saab treeningu aega selgelt vähendada, mis annab võimaluse kasutada suuremaid mudeleid, mis võiksid täpsuse osas olla paremad.

Viidatud kirjandus

[1] Goodfellow, I., Bengio, Y. ja Courville, A. Deep Learning. Cambridge, Massachusetts: The MIT Press, 2016.

[2] Filho, T., Song, H., Nieto, M. P., Rodriguez, R. S., Kull, M. ja Flach, P. Classifier Calibration: How to assess and improve predicted class probabilities: a survey (2021). (2023-05-09). https://arxiv.org/abs/2112.10327.

[3] Kängsepp, M. Calibration of Convolutional Networks. TÜ arvutiteaduse instituudi magistritöö, 2018.

[4] Arusoo, E. Tehisnärvivõrkude kirjeldamine väikese maailma omaduse abil. Tallinna Tehnikaülikooli tervisetehnoloogiate instituudi magistritöö, 2020.

[5] He, K., Zhang, X., Ren, S. ja Sun, J. Deep residual learning for image recognition (2015). (2023-05-09). https://arxiv.org/abs/1512.03385.

[6] Huang, G., Sun, Y., Liu, Z., Sedra, D. ja Weinberger, K. Deep networks with stochastic depth (2016). (2023-05-09). https://arxiv.org/abs/1603.09382.

[7] Zagoruyko, S. ja Komodakis, N. Wide residual networks (2015). (2023-05-09). https://arxiv.org/abs/1605.07146.

[8] Huang, G., Liu, Z., Maaten, L. van der ja Weinberger, K. Q. Densely connected convolutional networks (2018). (2023-05-09). https://arxiv.org/abs/1608.06993.

[9] Flach, P. A. Machine learning: the art and science of algorithms that make sense of data. Cambridge ; New York: Cambridge University Press, 2012.

[10] Kull, M., Nieto, M. P., Kängsepp, M., Filho, T., Song, H. ja Flach, P. Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with Dirichlet calibration (2019). (2023-05-09). https://arxiv.org/abs/1910.12656.

[11] Guo, C., Pleiss, G., Sun, Y. ja Weinberger, K. Q. On calibration of modern neural networks (2017). (2023-05-09). https://arxiv.org/abs/1706.04599.

[12] Naeini, M. P., Cooper, G. ja Hauskrecht, M. Obtaining Well Calibrated Probabilities Using Bayesian Binning. *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (veebruar 2015), lk. 2901–2907.

[13] Xiao, X., Mudiyanselage, T. B., Ji, C., Hu, J. ja Pan, Y. Fast Deep Learning Training through Intelligently Freezing Layers. *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (2019), lk. 1225–1232.

[14] Romero, M., Interian, Y., Solberg, T. ja Valdes, G. Targeted transfer learning to improve performance in small medical physics datasets. *Medical Physics* 47.12 (detsember 2020). arXiv:1912.06761, lk. 6246–6256.

[15] Chollet, F. Keras. https://keras.io. 2015.

[16] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images (2012).

[17] Netzer, Y, Wang, T, Coates, A, Bissacco, A, Wu, B ja Ng, A Y. Reading Digits in Natural Images with Unsupervised Feature Learning (2011). http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[18] Xiao, M, Rasul, K ja Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. https://arxiv.org/abs/1708.07747. (2023-05-09). 2017.

[19] HPC. https://hpc.ut.ee/. (2023-05-09).

[20] Kaggle. https://www.kaggle.com/. (2023-05-09).

[21] Li, W. cifar-10-cnn: Play deep learning with cifar datasets. https://github.com/BIGBALLON/cifar-10-cnn. (2023-05-09). 2017.

[22] Chen, L. Implementation of stochastic depth networks in keras. https://github.com/transcranial/stochastic-depth. (2023-05-09). 2016.

[23] Majumdar, S. Wide residual networks in keras. https://github.com/titu1994/Wide-Residual-Networks. (2023-05-09). 2018.

[24] Majumdar, S. Densenet implementation in keras. https://github.com/titu1994/DenseNet. (2023-05-09). 2018.

Lisad

Lisa 1. Repositoorium

Töös eksperimentide jooksutamiseks kirjutatud lähtekood on leitav aadressil https://github.com/OliverSavolainen/freezing-calibration.

Lisa 2. NLLi võrdlus mudelite ja andmestike kaupa kasutades järkjärguliste skeemide kalibreerimata tulemusi.

Mudel	Andmestik	Algne	Histo	Iso	Temp	Beeta	S1	S2	S1*	S2*
DenseNet 40	CIFAR-10	$0{,}4282_{7}$	$0{,}5725_{9}$	$0{,}2773_{3}$	$\mathbf{0{,}2251}_{1}$	$0{,}2392_{2}$	$0{,}4238_{6}$	$0{,}445_{8}$	$0{,}3832_{5}$	$0{,}3568_{4}$
DenseNet 40	CIFAR-100	$2{,}0174_{7}$	$4{,}1829_{9}$	$1{,}6491_{5}$	$\mathbf{1{,}0571}_{1}$	$1{,}1532_{2}$	$2{,}0277_{8}$	$1{,}9961_{6}$	$1{,}6168_{4}$	$1{,}4557_{3}$
ResNet 110	CIFAR-10	$0{,}3583_{7}$	$0{,}5472_{9}$	$0{,}2708_{3}$	$\mathbf{0{,}2093}_{1}$	$0{,}2139_{2}$	$0{,}3587_{8}$	$0{,}3202_{5}$	$0{,}3463_{6}$	$0{,}3065_{4}$
ResNet 110	CIFAR-100	$1{,}6937_{6}$	$4{,}2139_{9}$	$1{,}8926_{8}$	$\mathbf{1{,}0917}_{1}$	$1{,}1318_{2}$	$1{,}6912_{5}$	$1{,}7315_{7}$	$1{,}2053_{3}$	$1{,}248_{4}$
ResNet 110 (SD)	CIFAR-100	$1{,}3525_{7}$	$4{,}0187_{9}$	$1{,}6371_{8}$	$\mathbf{0{,}9421}_{1}$	$0{,}9643_{2}$	$1{,}2317_{5}$	$1{,}252_{6}$	$1{,}1423_{3}$	$1{,}1787_{4}$
ResNet 152 (SD)	SVHN	$0{,}0854_{3}$	$0{,}2887_{9}$	$0{,}1093_{6}$	$\mathbf{0{,}0786}_{1}$	$0{,}0796_{2}$	$0{,}1031_{4}$	$0{,}118_{7}$	$0{,}1063_{5}$	$0{,}1206_{8}$
Wide ResNet 32	CIFAR-10	$0{,}3817_{8}$	$0{,}5132_{9}$	$0{,}2327_{3}$	$\mathbf{0{,}1915}_{1}$	$0{,}202_{2}$	$0{,}3621_{7}$	$0{,}3408_{6}$	$0{,}306_{4}$	$0{,}3091_{5}$
Wide ResNet 32	CIFAR-100	$1{,}8022_{8}$	$3{,}9352_{9}$	$1{,}5607_{5}$	$\mathbf{0{,}9445}_{1}$	$1{,}0386_{2}$	$1{,}8008_{7}$	$1{,}7764_{6}$	$1{,}3175_{4}$	$1{,}2165_{3}$
Keskmine järk		$6{,}62_{8}$	$9{,}0_{9}$	$5{,}12_{5}$	$\mathbf{1{,}0}_{1}$	$2{,}0_{2}$	$6{,}25_{6}$	$6{,}38_{7}$	$4{,}25_{3}$	$4{,}38_{4}$

Lisa 3. ECE (%) võrdlus mudelite ja andmestike kaupa kasutades järkjärguliste skeemide kalibreerimata tulemusi

Model	Dataset	Uncal	Histo	Iso	Temp	Beeta	S1	S2	S1*	S2*
DenseNet 40	CIFAR-10	5.5_{7}	2.13_{4}	1.68_{2}	$\mathbf{0.95}_{1}$	1.7_{3}	5.58_{8}	5.8_{9}	5.17_{6}	5.02_{5}
DenseNet 40	CIFAR-100	21.16_{9}	11.97_{5}	5.25_{2}	$\mathbf{0.9}_{1}$	6.03_{3}	20.82_{7}	21.11_{8}	13.33_{6}	11.86_{4}
ResNet 110	CIFAR-10	4.75_{6}	1.25_{2}	1.47_{4}	$\mathbf{1.13}_{1}$	1.42_{3}	5.05_{9}	4.83_{7}	4.88_{8}	4.69_{5}
ResNet 110	CIFAR-100	18.48_{7}	9.06_{4}	6.54_{3}	$\mathbf{2.38}_{1}$	4.6_{2}	18.58_{8}	18.65_{9}	11.45_{5}	12.37_{6}
ResNet 110 (SD)	CIFAR-100	15.86_{9}	7.6_{4}	5.21_{3}	$\mathbf{1.21}_{1}$	3.55_{2}	13.91_{7}	14.26_{8}	12.43_{5}	13.25_{6}
ResNet 152 (SD)	SVHN	0.86_{5}	0.53_{3}	$\mathbf{0.25}_{1}$	0.61_{4}	0.5_{2}	1.66_{6}	1.95_{8}	1.94_{7}	2.33_{9}
Wide ResNet 32	CIFAR-10	4.51_{9}	1.01_{3}	1.19_{4}	$\mathbf{0.8}_{1}$	0.97_{2}	4.36_{6}	4.46_{7}	3.99_{5}	4.49_{8}
Wide ResNet 32	CIFAR-100	18.78_{9}	7.64_{4}	5.82_{2}	$\mathbf{0.78}_{1}$	7.01_{3}	18.38_{8}	18.22_{7}	12.64_{6}	9.76_{5}
Keskmine järk		7.62_{8}	3.62_{4}	2.62_{3}	$\mathbf{1.38}_{1}$	2.5_{2}	7.38_{7}	7.88_{9}	6.0_{5}	6.0_{5}

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Oliver Savolainen**,

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

   Konvolutsiooniliste neurovõrkude kalibreerimine järkjärgulise külmutamise meetodiga,

   mille juhendaja on Meelis Kull,

   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, kuni autoriõiguse kehtivuse lõppemiseni.

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

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Oliver Savolainen
09.05.2023