Parametritehokkaat hienosäätömenetelmät neuroverkoissa

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Joulukuu 2024
Otto-Julius Kyttänen

TURUN YLIOPISTO
Tietotekniikan laitos

Otto-Julius Kyttänen: Parametritehokkaat hienosäätömenetelmät neuroverkoissa

TkK-tutkielma, 20 s.
Tietotekniikka
Joulukuu 2024

Tutkielmassa tarkastellaan parametritehokkaita hienosäätömenetelmiä (engl. Parameter Efficient Fine-Tuning, PEFT), kuten LoRA ja QLoRA, joita hyödynnetään suurten neuroverkkojen hienosäädössä erityisesti luonnollisen kielen käsittelyn sovelluksissa. Tutkimuksen päätavoitteena on arvioida näiden menetelmien suorituskykyä ja resurssitehokkuutta verrattuna perinteiseen hienosäätöön. Tulokset osoittavat, että PEFT-menetelmät vähentävät merkittävästi laskennallisia resursseja ja muistinkulutusta, säilyttäen tai jopa parantaen mallien tarkkuutta. Menetelmien heikkouksina havaittiin kuitenkin matalan asteen pullonkaula ja kvantisoinnin rajoituksia, jotka vaikuttavat mallien tarkkuuteen erityisesti suurilla parametrimäärillä ja laajalla koulutusdatalla. Tutkielman löydökset korostavat PEFT-menetelmien potentiaalia tehdä mallien hienosäädöstä huomattavasti resurssitehokkaampaa, mutta herättävät myös kysymyksiä hienosäädön tulevaisuuden tarpeellisuudesta.

Asiasanat: parameteritehokas hienosäätö, LoRA, QLoRA, kvantisaatio, neuroverkot, optimointi

Sisällys

1 Johdanto

Neuroverkot ovat kehittyneet merkittävästi viimeisten vuosikymmenten aikana, ja niiden sovellukset ovat keskeisessä roolissa monissa tekoälyn sovelluksissa, kuten luonnollisen kielen käsittelyssä (engl. natural language processing, NLP), kuvatunnistuksessa ja robotiikassa. Neuroverkkojen teho perustuu niiden kykyyn löytää suurista datamassoista monimutkaisia rakenteita, mikä mahdollistaa mallien kehittämisen monenlaisiin tehtäviin.

Nykyiset neuroverkkopohjaiset perusmallit (engl. foundational model) pohjautuvat kahteen vaiheeseen. Ensin malli *esikoulutetaan* (engl. pre-training) laajalla määrällä dataa yksinkertaisiin tehtäviin, esimerkiksi ennustamaan puuttuva sana lauseessa (engl. Masked Language Modeling, MLM) tai luokittelemaan kuva oikeaan kategoriaan. Esikoulutus kehittää mallin kykyä ymmärtää kielen rakennetta tai kuvien geneerisiä kuvioita. Toisessa vaiheessa mallia *hienosäädetään* (engl. fine-tuning) tehtävään kohdistetulla datalla moniin erilaisiin jatkotehtäviin (engl. downstream tasks) [1][2]. Hienosäädöllä saadaan aikaan esimerkiksi, että malli oppii sekä seuraamaan ohjeita että vastaamaan kysymyksiin.

Suurimpien nykyaikaisten mallien, kuten NLP-mallien, parametrien määrät mitataan miljardeissa, ja niiden kouluttaminen vaatii huomattavan paljon laskenta-aikaa. Perinteinen hienosäätö tehdään mallin kaikkiin parametreihin, jolloin muutaman uuden asian opettaminen mallille voi viedä useita päiviä. Tämä lähestymistapa kuluttaa merkittävästi laskentatehoa, mikä lisää kustannuksia. Hienosäätöä nopeut-

tamaan on kehitetty useita parametritehokkaita menetelmiä. Näiden menetelmien käyttö voi joissain tapauksissa parantaa myös mallin tarkkuutta jo alentuneiden kustannusten ja laskenta-ajan lisäksi [2].

Tutkielma tekee katsauksen tunnetuimpiin *parametritehokkaisiin hienosäätömenetelmiin* (engl. Parameter Efficient Fine-Tuning, PEFT), jotka ovat laajalti käytössä luonnollisen kielen prosessoinnissa sekä laajenevasti muissa neuroverkkojen suuntauksissa, kuten konenäössä. Tutkielman tutkimuskysymykset ovat:

(TK1) Mitkä ovat nykyiset PEFT-menetelmät?

(TK2) Miten PEFT-menetelmät vertautuvat mallin suorituskyvyn ja resurssien kulutuksen suhteen?

(TK3) Mitkä ovat nykyisten PEFT-menetelmien heikkoudet ja miten niitä voidaan ratkaista?

Tutkielma toteutettiin pääasiassa kirjallisuuskatsauksena, mutta se sisältää myös empiirisiä kokeita tutkimuskysymykseen TK2 vastattaessa. Tutkimuksessa käytetyt tieteelliset lähteet löydettiin pääasiassa IEEE Xplore- ja Google Scholar -hakukannoista. Tiedonhaku suoritettiin käyttämällä hakusanoja, kuten “parameter-efficient fine-tuning”, “low-rank adaptation”, ja “quantification of large language models”.

Hakutulosten suuri määrä (usein satoja tai tuhansia julkaisuja) rajattiin otsikoiden ja tiivistelmien perusteella. Valintakriteereinä käytettiin tutkimuksen relevanssia sekä julkaisun yleistä laatua ja tunnettavuutta alalla. Aineiston valinnassa ensisijaisena tavoitteena oli käyttää vertaisarvioituja julkaisuja. Kuitenkin, aiheen ajankohtaisuuden ja nopeasti kehittyvän tutkimuskentän vuoksi, aineistoon sisällytettiin myös arXiv-preprint-artikkeleita, jotta tutkimus kattaa uusimmat löydökset.

Tutkielma on jaettu seuraaviin osioihin: Luvussa 2 esitellään neuroverkkojen toimintaperiaatteet, jotka muodostavat pohjan tutkittaville menetelmille. Luvussa 3 käsitellään parametritehokkaita hienosäätömenetelmiä LoRA ja QLoRA, sekä nii-

den heikkouksia ja rajoituksia. Luvussa 4 vertaillaan menetelmiä perinteiseen hienosäätöön empiiristen kokeiden avulla ja arvioidaan niiden suorituskykyä sekä resurssitehokkuutta. Luvussa 5 esitetään tutkimuksen johtopäätökset ja pohditaan tulevia tutkimusmahdollisuuksia.

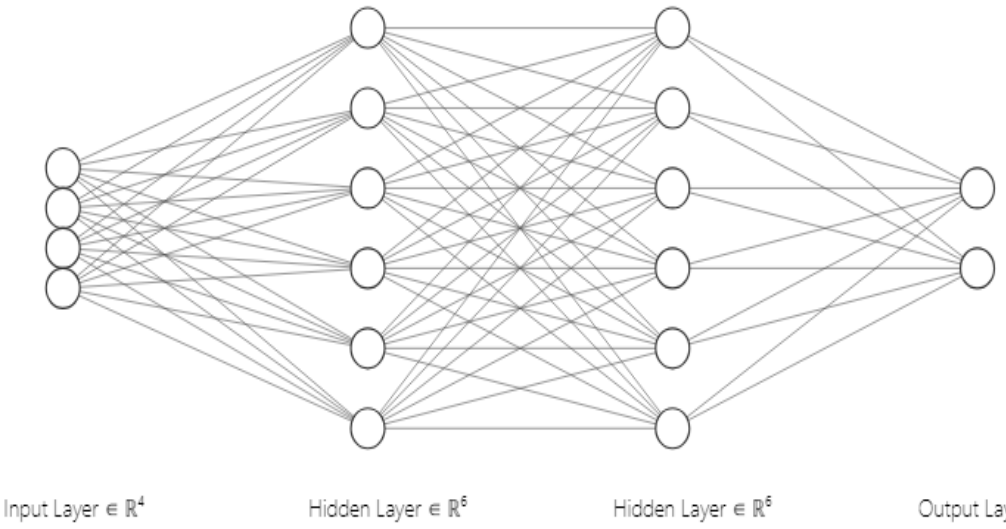
Empiirisen tutkimuksen osalta käytetyt mallit ja tutkimuksessa hyödynnetty koodi ovat julkisesti saatavilla osoitteesta `https://github.com/OttoKy/peft_tutkielma`, mikä varmistaa tulosten toistettavuuden.

2 Neuroverkot

Viimeisintä tekniikkaa edustavat (engl. state-of-the-art) tekoälymallit ovat lähes poikkeuksetta syviä neuroverkkoja. Syvät neuroverkot perustuvat *piirreoppimiseen* (engl. representation learning), jonka ansiosta verkkoihin voidaan syöttää raakaa dataa ilman, että piirteet pitäisi manuaalisesti suunnitella (engl. manual feature engineering) [3]. Kuten kuvassa 2.1 esitetään, neuroverkko rakentuu neuronikerroksista, jotka ovat:

- **sisääntulokerros**; jossa data syötetään verkkoon
- **piilokerrokset**; joissa dataa prosessoidaan
- **ulostulokerros**, joka tuottaa ennustuksen.

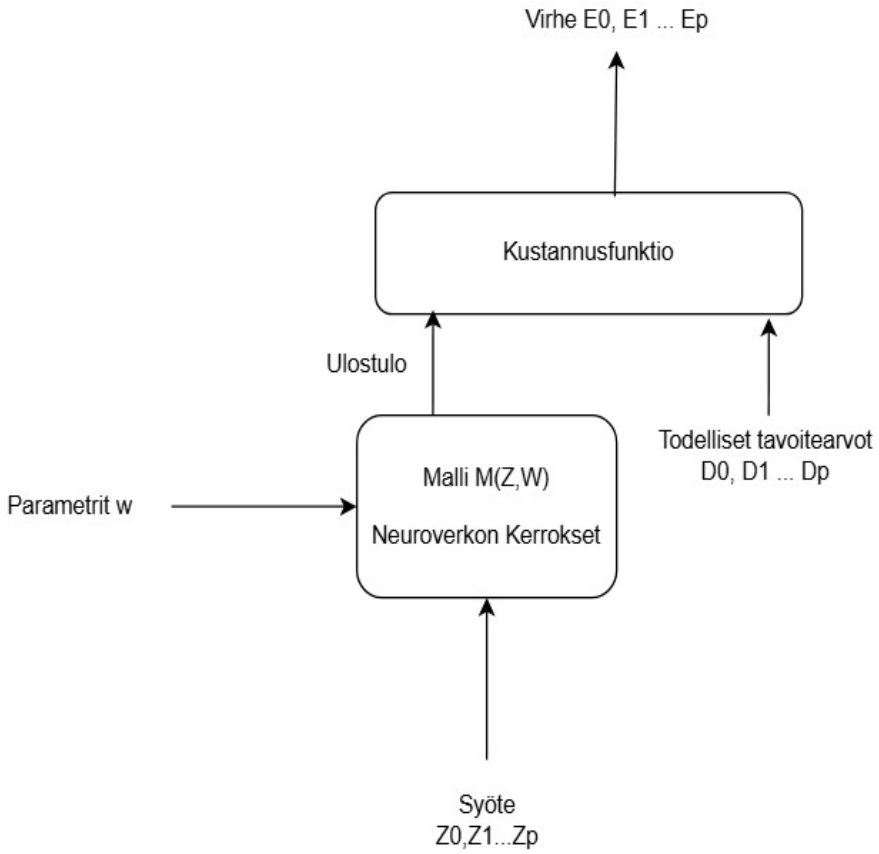
Jokainen kerros koostuu neuroneista, jotka vastaanottavat syötteen (engl. input) edelliseltä kerrokselta ja tuottavat ulostulon (engl. output) seuraavalle. Kuvan 2.1 esimerkissä eteenpäinkytketty neuroverkko sisältää 4 neuronia sisääntulokerroksessa, 6 neuronia kummassakin piilokerroksessa sekä 2 neuronia ulostulokerroksessa. Neuronien välillä olevat yhteydet ovat painoja, jotka oppimisprosessin aikana säätyvät. Neuroverkot jaetaan tyypillisesti kahteen päätyyppiin: eteenpäinkytkettyihin verkkoihin (engl. feedforward neural networks), joissa syöte kulkee vain eteenpäin, ja toistuviin verkkoihin (engl. recurrent neural networks), joissa tiedonsiirto voi olla kaksisuuntaista [4].



Kuva 2.1: Eteenpäinkytketyn neuroverkon rakenne. Verkossa on 4 sisääntuloneuronia, kaksi piilokerrosta, joissa kummassakin on 6 neuronia, sekä 2 ulostuloneuronia. Neuronien välillä olevat viivat edustavat painoja, jotka säädetään oppimisprosessin aikana.

Tyypillinen neuroverkon koulutusprosessi koostuu toistuvista kierroksista, joissa jokaisessa suoritetaan *eteenpäinheitto* (engl. forward propagation), eli syötetään dataa verkon läpi ja tuotetaan ennuste [4]. Syötettävä opetusdata koostuu syötearvoista sekä todellisista tavoitearvoista, joihin ennustetta verrataan. Jokaisen eteenpäinheiton lopussa lasketaan ennusteen ja todellisen arvon välinen ero, jota kutsutaan *virheeksi*. Lopuksi painoja muokataan *takaisinheiton* (engl. backpropagation) avulla siten, että virhe pienenee asteittain [5]. Mallin oppiminen perustuu siis virheen *minimointiin* ja se voidaan mallintaa optimointitehtävänä, jossa virhe muodostaa kustannusfunktion (engl. loss function) ja painot ovat muuttujia (ks. kuva 2.2). Seuraavaksi tarkastelemme yksityiskohtaisemmin, miten eteenpäinkytketyssä neuroverkossa ennustukset muodostuvat.

Neuronien välillä on yhteydet (engl. connections), joilla on omat painonsa w_i. Lisäksi piilokerrosten ja ulostulokerroksen neuroneilla on harhat (engl. bias) b. Kun-



Kuva 2.2: Havainnollistava kaavio kuinka malli laskee virheen kustannusfunktion avulla [5].

kin piilokerroksen ja ulostulokerroksen neuronin arvo muodostuu edellisen kerroksen neuronien arvoista x_i, yhteyksien painoista w_i sekä neuronin harhasta. Arvo lasketaan painotettuna summana, jossa n on edellisen kerroksen neuronien lukumäärä [3][6]:

$$z = \sum_{i=1}^{n} w_i x_i + b.$$

Ennen prosessin toistamista ja siirtymistä seuraavaan kerrokseen, painotettu summa syötetään aktivaatiofunktioon. Tämän tarkoitus on tuoda epälineaarisuutta malliin, mikä mahdollistaa monimutkaisempien suhteiden oppimisen syötteiden välillä. Yksi yleisesti käytetty aktivaatiofunktio on ReLU (engl. Rectified Linear Unit), joka on osoittautunut tehokkaaksi syvien neuroverkkojen aktivaatiofunktiona [3]. ReLU saa

syötteenään painotetun summan z, ja se määritellään seuraavasti:

$$a = \text{ReLU}(z) = \max(0, z).$$

Tämä tarkoittaa, että jos z on positiivinen, aktivoitunut arvo a säilyy ennallaan ($a = z$), ja jos z on negatiivinen, tulos on $a = 0$ [7].

Kun eteenpäinheitossa saavutetaan ulostulokerros, verkko tuottaa ennustuksen kerroksen neuroneissa. Tämä ennustus voi esimerkiksi määrittää, onko syötetty kuvio kissa vai koira. Ennustettua arvoa verrataan opetusdatan todelliseen arvoon *kustannusfunktion* avulla, joka mittaa ennusteen tarkkuutta [5][6]. Yksi yleisesti käytetty kustannusfunktio on keskimääräinen neliövirhe (engl. Mean Squared Error, MSE):

$$E = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \left(Y_i - \hat{Y}_i \right)^2,$$

missä E on kokonaiskustannus, n on datapisteiden lukumäärä, Y_i on todellinen arvo datapisteelle i, ja $\hat{Y}_i$ on mallin ennustus kyseiselle datapisteelle i. Ennustus $\hat{Y}_i$ saadaan ulostulokerroksen neuronien aktivoituneista arvoista a_i.

Kun kustannusfunktio on määritelty, verkon tavoitteena on minimoida sen arvo säätämällä painoja ja harha-termejä. Tämä minimointi toteutetaan iteratiivisessa optimointiprosessissa, jossa ensimmäisellä eteenpäinheitolla lasketaan verkon nykyinen kustannusfunktion arvo. Tämän jälkeen takaisinheittoprosessilla lasketaan kustannusfunktion osittaisderivaatat kaikkien painojen ja harha-termien suhteen. Nämä osittaisderivaatat muodostavat gradientin, joka osoittaa, miten kustannusfunktion arvo muuttuu painojen ja harha-termien muuttuessa. Gradienttien laskennassa hyödynnetään ketjusääntöä [6]:

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \frac{\partial E}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}},$$

$$\frac{\partial E}{\partial b_j^{(l)}} = \frac{\partial E}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}},$$

missä l on kerroksen indeksi, $w_{ij}^{(l)}$ on paino neuronien i ja j välillä kerroksessa l, sekä $z_j^{(l)}$ on neuroni j:n painotettu summa kerroksessa l.

Gradienttien laskennan jälkeen optimointialgoritmi, kuten *stokastinen jyrkimmän laskun menetelmä* (engl. Stochastic Gradient Descent, SGD), käyttää näitä gradientteja päivittääkseen painoja ja harha-termejä. Gradienttipohjaiset menetelmät ovat hyvin yleisiä oppimismenetelmiä syväoppivissa neuroverkoissa niiden kustannustehokkuuden ja skaalautuvuuden ansiosta [8].

SGD päivittää painoja ja harha-termejä seuraavasti:

$$w \leftarrow w - \eta \cdot \frac{\partial E}{\partial w}, \quad b \leftarrow b - \eta \cdot \frac{\partial E}{\partial b},$$

missä $\frac{\partial E}{\partial w}$ ja $\frac{\partial E}{\partial b}$ ovat kustannusfunktion gradientit painojen ja harhojen suhteen, ja η on oppimisnopeus, joka määrittää muutoksen suuruuden. Tätä prosessia toistetaan, kunnes saavutetaan ennalta määritelty lopetuskriteeri, kuten gradientin normin $||\nabla E||$ pieneneminen alle tietyn kynnyksen ϵ, tai kun virhe ei enää pienene merkittävästi.

Toistamalla tätä iteratiivista optimointiprosessia verkko oppii vähitellen datan piirteitä. Päivittämällä painoja ja harha-termejä gradienttien perusteella verkko minimoi virheen mallin ennustuksen ja todellisen vastauksen välillä, mikä johtaa malliin, joka kykenee yleistämään opetusdatan ulkopuolelle.

3 Hienosäätömenetelmät

Neuroverkon hienosäädöllä tarkoitetaan sitä, että jo olemassa olevalle perusmallille opetetaan jotain uutta tai parannetaan jo olemassa olevaa osaamista. Tämä vaatii yleensä dataa, joissa on tuhansia esimerkkejä halutusta aiheesta [9]. Yhdestä esikoulutetusta perusmallista voidaan siis kouluttaa monta eri mallia, jotka ovat kaikki erikoistuneet eri tehtäviin, esimerkiksi vaikka ohjelmointiin tai kirjoittamiseen. Perinteinen hienosäätö tehdään mallin kaikkiin parametereihin, mikä on laskennallisesti todella työlästä. Tämän takia on kehitetty tehokkaampia menetelmiä, jotka esitellään tässä luvussa.

3.1 LoRA

LoRA (engl. Low-Rank Adaptation) on tehokas PEFT-menetelmä, jolla parametrien määrää saadaan vähennettyä huomattavasti hienosäädön aikana. Se on suunniteltu neuroverkkoihin, missä kaikki kerrokset ovat täysin kytkettyjä toisiinsa (engl. dense). LoRAn tehokkuus perustuu siihen, että esikoulutetuilla malleilla katsotaan olevan todella pieni luontainen ulottuvuus (engl. intrinsic dimension) [10]. Tämä tarkoittaa, että mallin oppimiskykyä ei määrää sen parametrien kokonaismäärä, vaan ainoastaan niiden parametrien joukko, jotka ovat välttämättömiä uuden tiedon omaksumiseen. Vaikka esikoulutettu malli sisältäisi miljardeja parametreja, vain pieni osa niistä on aktiivisesti tarpeen jokaisen uuden tehtävän oppimisessa. Tämä johtuu siitä, että esikoulutettu malli on jo oppinut yleisiä ominaisuuksia ja rakenteita laajoista

tietomassoista.

Esikoulutetun mallin kerros voidaan näyttää muodossa:

$$W_0 \in \mathbb{R}^{d \times k},$$

missä W_0 on esikoulutetun mallin alkuperäinen painomatriisi, d on sisääntulon ulottuvuus (engl. input dimension) ja k on ulostulon ulottuvuus (engl. output dimension).

Kun mallin hienosäätö aloitetaan, matriisi W_0 jäädytetään ja pidetään muuttumattomana koko prosessin ajan. Matriisin W_0 pohjalta luodaan kaksi uutta matriisia A ja B. Näiden matriisien ulottuvuudet muodostuvat alkuperäisestä matriisista W_0, sekä hienosäätöön valitusta arvosta r, jonka pitää olla huomattavasti pienempi kuin $\min(d, k)$ [2]:

$$A \in \mathbb{R}^{r \times k}, \qquad B \in \mathbb{R}^{d \times r}.$$

Nämä matriisit muodostavat yhdessä matriisin ΔW, joka on approksimaatio kerroksen kaikkien parametrien W_0 päivittämisestä:

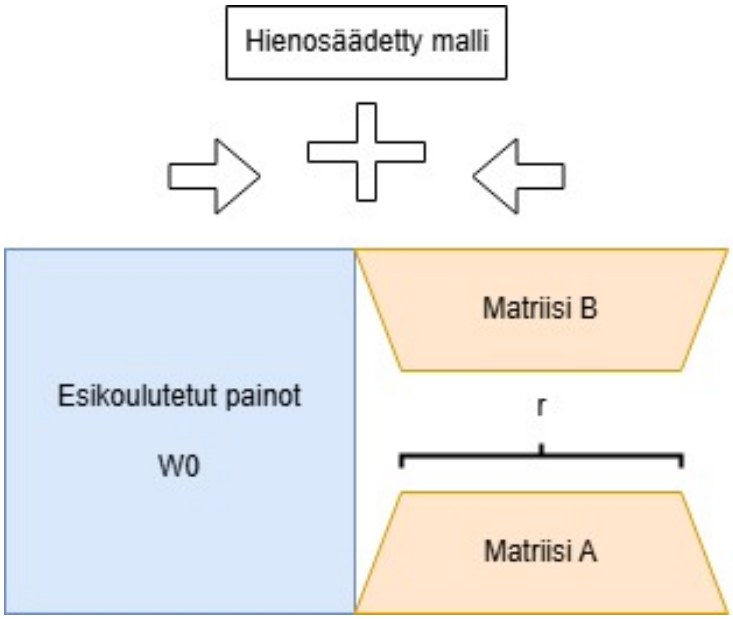
$$\Delta W = BA.$$

Hienosäädön jälkeen lopputulos kerroksen matriisille on:

$$W = W_0 + \Delta W.$$

Matriisia W kutsutaan LoRA-adapteriksi. LoRA-adapteri lisätään perusmallin valittuihin kerroksiin, joita tässä kutsutaan kohdemoduuleiksi. LoRA-adapteri muokkaa kohdemoduulin painomatriiseja siten, että mallin käyttäytyminen mukautuu haluttuun tehtävään sopivaksi. Nämä adapterit eivät vaadi muutoksia mallin varsi-

naiseen arkkitehtuuriin, vaan ne lisätään suoraan olemassa oleviin kerroksiin pienikokoisina matriiseina. Kuvassa 3.1 on havainnollistettu, kuinka valmis malli syntyy jäädytettyjen painojen W_0, sekä matriisien AB yhteenlaskun tuloksena.



Kuva 3.1: Matriisien A ja B tulo ΔW lisätään valittuihin kerroksiin, jolloin syntyy uusi hienosäädetty malli [2].

LoRA vähentää hienosäädössä merkittävästi muistin tarvetta sekä vaadittavaa tallennustilaa. Menetelmän testauksen yhteydessä on onnistuttu vähentämään VRA-Min (engl. video random-access memory) käyttöä jopa 2/3, koska se ei vaadi jäädytettyjen parametrien optimointitilojen tallentamista [2]. LoRA-hienosäätö on myös ajallisesti nopeampaa perinteiseen hienosäätöön verrattuna, koska suurimalle osalle parametreistä ei tarvitse laskea gradientteja.

3.2 QLoRA

QLoRA (engl. Quantization Low-Rank Adaptation) yhdistää kaksi tehokasta keinoa säästää muistia ja parantaa mallien resurssitehokkuutta hienosäädön aikana. Nämä ovat kvantisointi (engl. quantization) ja LoRA. Kvantisointi on pitkään ollut

keskeinen tekniikka erityisesti mallien käyttövaiheessa (engl. inference). Sen avulla vähennetään laskennallista monimutkaisuutta pienentämällä painojen ja aktivaatioiden bittipituutta. QLoRA kykenee hyödyntämään tätä ominaisuutta myös mallin hienosäädön aikana, mikä tuo huomattavasti säästöä VRAM-kulutuksessa ilman, että mallin suorituskyky huononee merkittävästi [11].

Kvantisointi on tekniikka, jossa jatkuvat reaalilukuarvot muunnetaan rajoitetuksi joukoksi diskreettejä arvoja. Tämä tarkoittaa esimerkiksi, että luvut 3.2, 5.6 ja 7.9 voidaan kvantisoida lähimpiin kokonaislukuihin 3, 6 ja 8. Kvantisoidut arvot pienentävät lukujen tallennukseen tarvittavaa muistia ja vähentävät laskennan monimutkaisuutta [11]. Tämä prosessi on tärkeä isojen neuroverkkojen kohdalla, sillä se vähentää merkittävästi tarvittavan muistin määrää [12].

Neuroverkoissa tyypillisesti käytetyt paino- ja aktivaatioarvot mallin koulutuksessa ovat 16- tai 32-bittisiä liukulukuja. Kvantisointitekniikasta riippuen nämä liukuluvut muutetaan matalampaan bittitarkkuuteen, esimerkiksi 4- tai 8-bittisiksi kokonaisluvuiksi. QLoRA yhdistää kaksi eri kvantisointitekniikkaa: 4-bittinen NormalFloat-kvantisointi (NF4) ja kaksoiskvantisointi (engl. Double Quantization, DQ). Double Quantization tarkoittaa kahden tason kvantisointimenetelmää, jossa myös kvantisoinnissa käytettävät skaalausarvot muutetaan matalampaan bittitarkkuuteen [11]. Skaalausarvot toimivat painojen ja aktivaatioiden kvantisoinnissa korjauskertoimina, jotka mahdollistavat alkuperäisten arvojen tarkemman palauttamisen matalasta bittitarkkuudesta. Lisäksi QLoRA käyttää 16-bittistä BrainFloat-tietotyyppiä, joka on yleisesti käytössä neuroverkkojen koulutuksessa, sillä se mahdollistaa pienemmän muistinkulutuksen säilyttäen laskennallisen tarkkuuden.

QLoRA:n ja LoRA:n ero on mallin painojen bittipituudessa ja niiden käsittelytavassa. Kun alkuperäisen mallin painot jäädytetään, QLoRA muuntaa nämä parametrit NF4-tietotyyppiin. Tällöin LoRA-parametrit, jotka valitaan alkuperäisistä

painoista, muunnetaan 16-bittiseen BrainFloat-tietotyyppiin. Sekä eteen- että taaksepäin heittojen aikana mallin jäädytetyt parametrit dekvantisoidaan väliaikaisesti 16-bittiseen muotoon, jotta tarvittavat laskutoimitukset voidaan suorittaa [11].

QLoRA:n avulla voidaan hienosäätää perusmalleja jopa 16 kertaa pienemmällä muistimäärällä verrattuna perinteiseen hienosäätöön [11]. Tämä on merkittävää esimerkiksi suurten kielimallien hienosäädössä, sillä nyt jopa suurimmat julkisesti saatavilla olevat mallit voidaan hienosäätää yhdellä GPU:lla (Graphics Processing Unit).

3.3 Menetelmien rajoitukset ja heikkoudet

Vaikka PEFT-menetelmien avulla saavutetaan usein huipputuloksia eri suorituskykymittareilla (engl. benchmark), on niissä silti omat rajoituksensa. Lisäksi mallin menestyminen näillä mittareilla ei suoraan tarkoita, että se kykenisi yleistämään yhtä hyvin käytännön sovelluksissa tai erilaisissa ympäristöissä. Mallien tarkkuus on hyvin paljon kiinni siitä, miten lähellä hienosäätöön käytetty koulutusdata on testaukseen käytettyä dataa [11]. Lisäksi PEFT-menetelmät ovat tällä hetkellä vahvasti sidoksissa muunnin-arkkitehtuuriin (engl. transformers), mikä rajoittaa niiden sovellettavuutta tehtäviin, joissa tarvitaan muita neuroverkkotyyppejä [13].

Perinteiseen hienosäätöön verrattuna PEFT-menetelmien heikkoudet tulevat usein esiin tarkkuudessa. Vaikka alkuperäiset tutkimukset esittelevät PEFT-menetelmien suoriutuvan erinomaisesti useilla mittareilla, jatkotutkimukset ovat tuoneet esiin niiden rajoituksia [2][11][14][15]. PEFT-menetelmät saavuttavat kyllä hyvän suorituskyvyn tietyissä sovellustehtävissä, mutta täyden hienosäädön ja PEFT-menetelmien välillä on yhä merkittävä ero tarkkuudessa vaativilla tehtäväalueilla, kuten matemaattisessa päättelyssä ja ohjelmoinnissa.

Nämä suorituskykyerot johtuvat pääosin kahdesta keskeisestä tekijästä: matalan asteen pullonkaulasta (engl. low-rank bottleneck) ja tarkkuus- ja kvantisointirajoi-

tuksista. Matalan asteen r päivitykset tekevät LoRA:sta parametritehokkaan, mutta samalla ne rajoittavat suurten kielimallien kykyä omaksua ja yleistää tietoa alaspesifisissä tehtävissä. Tämä rajoitus heikentää LoRA:n suorituskykyä erityisesti tietoa ja taitoa vaativilla alueilla [14][15]. Tämä ei kuitenkaan tarkoita, että suurempi r tarkoittaisi automaattisesti parempaa tarkkuutta. LoRA on myös herkkä hyperparametrien, kuten oppimisnopeuden, kohdemoduulien, rank-arvon sekä skaalauskertoimien valinnalle. Näiden optimointi on välttämätöntä, jotta mallin suorituskyky voisi edes yltää perinteisen hienosäädön tarkkuustasolle [14].

Lisäksi QLoRA-menetelmässä on havaittu, että vaikka menetelmä saavutti huipputuloksia pienemmillä malleilla, sen tarkkuus jää yleensä jälkeen perinteisestä hienosäädöstä suurilla, kuten 33 ja 65 miljardin parametrin malleilla [11]. Uudemmat tutkimukset ovat osoittaneet, että mallin koko ja siihen liittyvä koulutusdata vaikuttavat kvantisoinnin tehokkuuteen [16]. Nämä havainnot perustuvat kvantisointiin, joka on suoritettu joko esikoulutuksen (engl. pre-training) tai sen jälkeisen vaiheen (engl. post-training) yhteydessä, mutta todennäköisesti tulokset olisivat samansuuntaisia myös hienosäädön tapauksessa, erityisesti suurilla malleilla ja suurilla datamäärillä. Tämä tarkkuuden heikkeneminen johtuu siitä, että laskentatarkkuuden pienentäminen vähentää mallin tehokasta parametrimäärää (engl. effective parameter count) [16].

PEFT-menetelmien käyttö on monessa tapauksessa välttämätöntä, kun perinteinen hienosäätö muodostuu käytännössä liian raskaaksi. Tämä ei kuitenkaan tarkoita, että PEFT-menetelmät olisivat täysin vailla resurssivaatimuksia. Esimerkiksi suurien kielimallien hienosäädön vaatimat resurssit kasvavat eksponentiaalisesti mallin koon ja koulutusdatan lisääntyessä. Toisaalta suuret kielimallit kykenevät jo nyt oppimaan tehtäviä pelkästään vähäisten esimerkkien avulla, ilman että niiden parametreja tarvitsee hienosäätää [9]. Tämä ns. *in-context learning* -ominaisuus mahdollistaa mallin soveltamisen moniin tehtäviin pelkästään sopivien syötteiden avul-

la, mikä voi tulevaisuudessa vähentää hienosäätötarvetta entisestään. Tämän lisäksi uudemmat menetelmät, kuten *Chain-of-Thought* (CoT), voivat tarjota tehokkaamman lähestymistavan hienosäädön korvaamiseen [17]. CoT-menetelmät hyödyntävät mallien sisäistä päättelykykyä monimutkaisten ongelmien ratkaisemiseen lisäämällä syötteisiin ohjeita, jotka auttavat mallia jakamaan ongelman osiin. Tämä vähentää tarvetta muokata mallin parametreja tehtäväkohtaisesti, mikä voi tarjota pitkäaikaisen ratkaisun hienosäätöön liittyviin resurssiongelmiin.

4 Hienosäätömenetelmien vertailu

Tässä luvussa vertaillaan täyden hienosäädön menetelmää sekä PEFT-menetelmiä LoRA ja QLoRA. Vertailun tavoitteena on selvittää ja vastata tutkimuskysymykseen siitä, miten PEFT-menetelmät vertautuvat mallin suorituskyvyn ja resurssien kulutuksen suhteen. Empiirisessä kokeessa hienosäädettiin Metan kehittämää Llama-3.2-1B-perusmallia AI2 ARC-easy -aineistolla, joka sisältää 2250 esimerkkiä peruskoulutason tiedekysymyksistä [18][19]. Hienosäädön aikana mitattiin koulutusaika sekä varatun GPU-muistin (VRAM) määrä. Mallien tarkkuutta arvioitiin lisäksi aineiston testiosalla, joka sisälsi 2380 mallille ennestään tuntematonta kysymystä.

4.1 Koulutusasetelma

Mallien hienosäätö perustuu osittain QLoRA-paperin koulutusasetelmiin [11]. Kaikki mallit hienosäädettiin ohjattuna oppimisena (engl. supervised learning) cross-entropy loss -tappiofunktiolla. Aineistossa oli selkeä ero kysymysten ja vastausten välillä, ja tästä syystä hienosäätö kohdistettiin vain vastauksiin, jotta malli keskittyy vain oikean vastauksen (a, b, c, d) ennustamiseen. Mallien oppimisnopeus η oli 2e-4, ja optimoijana käytettiin AdamW8bit-algoritmia [20]. Koulutus kesti kolme epookkia (engl. epoch), ja eräkoko (engl. batch size) oli 8. LoRA:n r-arvo oli 16, ja hienosäätö kohdistettiin seuraaviin kohdemoduuleihin: $W_q, W_k, W_v, W_o, W_{\text{gate}}, W_{\text{up}}, W_{\text{down}}$. Nämä moduulit edustavat mallin eri painomatriiseja, jotka liittyvät sen itseohjautuvaan mekanismiin (engl. attention modules) ja syväkerrosten toimintaan [21]. Hie-

nosäädössä käytettiin NVIDIA A100-SMX4-40GB -näytönohjainta. Tutkimuksessa käytetyt mallit ja koodi ovat julkisesti saatavilla osoitteessa `https://github.com/OttoKy/peft_tutkielma`, mikä varmistaa tulosten toistettavuuden.

4.2 Tulosten analysointi

Tuloksien perusteella voidaan huomata merkittäviä eroja tarkkuudessa, muistinkulutuksessa ja koulutusajassa (ks. taulukko 4.1) Tämä toimii lähtökohtana tarkemmalle analyysille eri hienosäätömenetelmien vahvuuksista ja heikkouksista, sekä siitä, mistä tulokset johtuvat.

Taulukko 4.1: Hienosäätömenetelmien suorituskykyvertailu, perinteinen hienosäätö, LoRA ja QLoRA

Malli ja tekniikka	Koulutusaika	Varattu muisti (GB)	Tarkkuus (%)
LLaMA 3.2-1B (PH)	**4 min 38 sek**	17.986	24,20
LLaMA 3.2-1B (LoRA)	7 min 5 sek	3.852	**76,09**
LLaMA 3.2-1B (QLoRA)	7 min 56 sek	**2.514**	75,59

Koulutusajassa täysi hienosäätö osoittautui nopeimmaksi menetelmäksi. Yleisesti PEFT-menetelmien oletetaan lyhentävän myös koulutusaikaa [11][22]. Tässä tapauksessa todennäköisesti aineiston ja perusmallin pieni koko muuttaa asetelmaa: koska koulutusaika on valmiiksi erittäin lyhyt, LoRA:n ja QLoRA:n käyttämät lisämatemaattiset operaatiot, voivat aiheuttaa suhteellisesti suurempaa hidastusta pienissä malleissa.

Muistinkulutus seuraa monia aikaisempia tutkimuksia kuten [2][11][22]. LoRA vähensi täyteen hienosäätöön verrattuna vaadittavaa muistia 76% ja QLoRA jopa 86%. Pienempi muistinkulutus on merkittävää monelta kannalta, kuten luvussa 3.2 käsiteltiin, erityisesti kun mallin tarkkuus ei pelkästään pysy linjassa täyden hienosäädön kanssa, vaan tässä tapauksessa jopa ylittää sen.

Ennustuksen tarkkuudessa PEFT-menetelmät osoittautuivat huomattavasti paremmiksi kuin täysi hienosäätö. Vaikka aikaisemmissakin tutkimuksissa PEFT-menetelmien on havaittu olevan täyttä hienosäätöä tarkempia tietyissä tapauksissa [2][11], on tässä tutkimuksessa havaittu ero poikkeuksellisen suuri. Todennäköinen syy täyden hienosäädön heikolle suorituskyvylle on ylisovittaminen (engl. overfitting), sillä koulutusaineisto oli pieni eikä hyperparametrejä optimoitu. Ylisovittamisessa malli oppii liiaksi koulutusaineiston erityispiirteitä, mikä käytännössä tarkoittaa, että se "muistaa" aineiston ulkoa [23]. Tämä heikentää mallin suorituskykyä testiaineistolla, jota malli ei ole aikaisemmin nähnyt, kuten tuloksista taulukossa 4.1 voidaan havaita.

Tehty tutkimus oli varsin rajattu. Vaikka tulokset pääosin vastaavat aiempien tutkimusten havaintoja, yleistettävyyttä voisi parantaa laajentamalla tutkimusta eri aineistoilla sekä perusmalleilla. Hyperparametrien, kuten oppimisnopeuden ja koulutusajan, optimointi esimerkiksi hyperparametrien haun avulla voisi johtaa luotettavampiin tuloksiin. Myös LoRA r-arvon ja kohdemoduulien tutkiminen olisi varmasti hyödyllistä.

5 Yhteenveto

Tutkielman tavoitteena oli vastata kolmeen keskeiseen tutkimuskysymykseen: (TK1) Mitkä ovat nykyiset PEFT-menetelmät? (TK2) Miten PEFT-menetelmät vertautuvat mallin suorituskyvyn ja resurssien kulutuksen suhteen? (TK3) Mitkä ovat nykyisten PEFT-menetelmien heikkoudet ja miten niitä voidaan ratkaista?

Hienosäätöä ja PEFT-tekniikoita pohjustettiin yleisellä kuvauksella neuroverkkojen toiminnasta luvussa 2. Tutkimuskysymykseen TK1 vastattiin luvussa 3: nykysin yleisimmin käytössä olevat PEFT-menetelmät ovat LoRA ja QLoRA. Näiden menetelmien suosio perustuu niiden tehokkuuteen laajojen kielimallien säätämisessä. Kielimallit ovat parametrikooltaan suurimpia neuroverkkoja.

Luku 3 vastaa myös tutkimuskysymykseen TK3 käsittelemällä PEFT-menetelmien rajoituksia. Näissä tekniikoissa on kaksi pääasiallista rajoitusta: matalan asteen pullonkaula ja kvantisointi. Matalan asteen pullonkaula viittaa siihen, että matriisien ulottuvuuden pienentäminen voi rajoittaa mallin kykyä oppia monimutkaisia riippuvuuksia. Kvantisointi puolestaan vähentää parametrien tarkkuutta, mikä saattaa johtaa informaation häviämiseen. Näiden rajoitusten vuoksi PEFT-menetelmät eivät pärjää täyden hienosäädön menetelmälle tehtäväalueilla, jotka vaativat syvää päättelyä ja laajaa tietämystä, kuten matemaattisessa päättelyssä.

Tutkimuskysymykseen TK2 saatiin vastaus empiirisellä kokeella luvussa 4. Nämä kokeet osoittivat PEFT-menetelmien olevan täysin ylivoimaisia resurssien kulutuksen suhteen verrattuna perinteiseen hienosäätöön. Tutkimuksesta kävi myös ilmi,

että PEFT-hienosäädetyt mallit pärjäsivät hyvin tarkkuudessa, kun koulutusaineisto oli ennen näkemättömän testiaineiston kanssa saman kaltaista. Nämä tulokset vastasivat kirjallisuudessa esitettyjä tuloksia.

PEFT-menetelmät ovat vakiinnuttaneet asemansa hienosäädössä. Näiden menetelmien hyödyt ovat kiistattomat nykyisessä ympäristössä, jossa vain harvalla organisaatiolla on varaa tai aikaa kouluttaa isoja perusmalleja. PEFT-menetelmät mahdollistavat suurten kielimallien tehokkaan hienosäädön huomattavasti pienemmillä laskennallisilla resursseilla, mikä avaa ovia laajemmalle tutkimus- ja sovellusjoukolle.

Resurssitehokkuus säilyy tekoälytutkimuksen keskeisenä tavoitteena myös tulevaisuudessa. Tutkielma herättää pohtimaan sitä, onko järkevämpää investoida yhteen suureen malliin, joka pystyy suorittamaan monia tehtäviä mutta on kallis kouluttaa ja ylläpitää, vai hyödyntää useita pienempiä, tehtäväkohtaisesti hienosäädettyjä malleja. Useiden pienempien mallien käyttö voi olla kustannustehokkaampaa ja mahdollistaa paremman suorituskyvyn tietyissä tehtävissä. Toisaalta jatkuva kehitys tarjoaa vaihtoehtoisia lähestymistapoja, kuten perusmallien syötteiden optimoinnin, mikä mahdollistaa parempien tulosten saavuttamisen ilman raskasta hienosäätöä ja tehtäväkohtaisia laajoja aineistoja.

Lähdeluettelo

[1] Llama Team, AI Meta, "The Llama 3 Herd of Models", *arXiv preprint*, 2024, DOI: 10.48550/arXiv:2407.21783.

[2] E. Hu, Y. Shen, P. Wallis et al., "LoRA: Low-Rank Adaptation of Large Language Models", *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022, DOI: 10.48550/arXiv.2106.09685.

[3] Y. LeCun, Y. Bengio ja G. Hinton, "Deep Learning", *Nature*, vol. 521: s. 436–444, 2015, DOI: 10.1038/nature14539.

[4] I. Goodfellow, Y. Bengio ja A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org, 30. marraskuuta 2024.

[5] Y. LeCun, L. Bottou, G. B. Orr ja K.-R. Müller, "Efficient BackProp", *Lecture Notes in Computer Science*, 1998, DOI: 10.1007/3-540-49430-8_2.

[6] D. E. Rumelhart, G. E. Hinton ja R. J. Williams, "Learning Representations by Back-Propagating Errors", *Nature*, vol. 323: s. 533–536, 1986, DOI: 10.1038/323533a0.

[7] A. F. M. Agarap, "Deep Learning using Rectified Linear Units (ReLU)", *arXiv preprint*, 2018, DOI: 10.48550/arXiv.1803.08375.

[8] D. P. Kingma ja J. Ba, "Adam: A Method for Stochastic Optimization", *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015, DOI: 10.48550/arXiv:1412.6980.

[9] T. B. Brown, B. Mann, N. Ryder et al., ”Language Models are Few-Shot Learners”, *Advances in Neural Information Processing Systems*: s. 1877–1901, 2020, DOI: 10.5555/3495724.3495883.

[10] A. Aghajanyan, S. Gupta ja L. Zettlemoyer, ”Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning”, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*: s. 7319–7328, 2021, DOI: 10.18653/v1/2021.acl-long.568.

[11] T. Dettmers, A. Pagnoni, A. Holtzman ja L. Zettlemoyer, ”QLoRA: Efficient Finetuning of Quantized LLMs”, *Advances in Neural Information Processing Systems 37 (NeurIPS 2023)*, 2023, DOI: 10.5555/3666122.3666563.

[12] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney ja K. Keutzer, ”A Survey of Quantization Methods for Efficient Neural Network Inference”, *Low-Power Computer Vision, Chapman and Hall/CRC*: s. 291–326, 2022, DOI: 10.1201/9781003162810-13.

[13] M. Yoshimura, T. Hayashi ja Y. Maeda, ”MAMBAPEFT: Exploring Parameter-Efficient Fine-Tuning for Mamba”, *arXiv preprint*, 2024, DOI: 10.48550/arXiv:2411.03855.

[14] D. Biderman, J. Portes, J. J. G. Ortiz et al., ”LoRA Learns Less and Forgets Less”, *Transactions on Machine Learning Research (TMLR)*, 2024, DOI: 10.48550/arXiv:2405.09673.

[15] Y. Mao, Y. Ge, Y. Fan et al., ”A Survey on LoRA of Large Language Models”, *Higher Education Press*, 2024, DOI: 10.1007/s11704-024-40663-9.

[16] T. Kumar, Z. Ankner, B. F. Spector et al., ”Scaling Laws for Precision”, *arXiv preprint*, 2024, DOI: 10.48550/arXiv:2411.04330.

[17] J. Wei, X. Wang, D. Schuurmans et al., ”Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”, *Advances in Neural Information Processing Systems*: s. 24 824–24 837, 2022, DOI: 10.5555/3600270.3602070.

[18] P. Clark, I. Cowhey, O. Etzioni et al., ”Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge”, *arXiv preprint*, 2018, DOI: 10.48550/arXiv.1803.05457.

[19] Meta AI, ”Llama 3.2: A 1B-Parameter Model”, 2024, url: https://huggingface.co/meta-llama/Llama-3.2-1B, 30. marraskuuta 2024.

[20] T. Dettmers, M. Lewis, Y. Belkada ja L. Zettlemoyer, ”8-bit Optimizers via Block-wise Quantization”, *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022, DOI: arXiv:2110.02861.

[21] A. Vaswani, N. Shazeer, N. Parmar et al., ”Attention Is All You Need”, *Advances in Neural Information Processing Systems*: s. 6000–6010, 2017, DOI: 10.5555/3295222.3295349.

[22] L. Lin, H. Fan, Z. Zhang, Y. Wang, Y. Xu ja H. Ling, ”Tracking Meets LoRA: Faster Training, Larger Model, Stronger Performance”, *arXiv preprint*, 2024, DOI: 10.48550/arXiv:2403.05231.

[23] S. Salman ja X. Liu, ”Overfitting Mechanism and Avoidance in Deep Neural Networks”, *arXiv preprint*, 2019, DOI: 10.48550/arXiv:1901.06566.