Toteutin ohjelman, jonka avulla voi luoda ja kouluttaa neuroverkkoja tunnistamaan MNIST-tietokannan käsinkirjoitettuja numeroita. Lisäksi ohjelma pitää kirjaa siitä, kuinka hyvin kukin verkko tunnistaa erilliseen 10000 kuvan testidatasettiin kuuluvat kuvat, mikä käytännössä toimii jokaisen verkon suorituskyvyn mittarina. Koulutukseen käytetään 60000 kuvan datasettiä. Kuvien luonteesta johtuen käytettävien verkkojen sisään- ja ulostulokerrosten koot ovat kiinteät, omassa ohjelmassani verkon sisääntulokerroksessa on 784 neuronia, mikä vastaa MNIST-datasetin 28x28 pikselin kuvakokoa ja vastaavasti ulostulokerroksen koko on 10, missä kukin ulostulovektorin alkio vastaa tiettyy numeroon 0-9 liittyvää luokittelutodennäköisyyttä. Lisäksi verkossa on yksi piilokerros, jonka koon käyttäjä määrittää ohjelmaa suoritettaessa. Ohjelman käyttöliittymä on tekstipohjainen.

Kuten neuroverkoissa yleensäkin, koulutuksessa käytetään vastavirta-algoritmia, jonka aikavaativuus riippuu lonnollisesti verkon koosta. Omassa ohjelmassani verkossa on kolme kerrosta. Jos piilokerroksessa on N neuronia, niin tällöin painokerroinmatriiseissa on 784*N ja 10*N alkiota. Verkkoa koulutettaessa prosessoidaan ensin syöte ja tämän jälkeen päivitetään painokertoimet. Syötteen prosessoinnissa suoritetaan yksi matriisi-vektoritulo, jonka aikavaativuus skaalautuu matriisin alkioiden mukaan. Näin ollen koko algoritmin aikavaativuus yhdelle kuvalle on O(N). Luonollisesti matriisien alkiot sekä vastavirta-algoritmin aikana laskettavat gradientit tulee pitää muistissa, joten tämä on myös yksittäiselle kuvalle pätevä tilavaativuus. Verkkoa koulutettaessa kuluva aika toki skaalautuu myös koulutusesimerkkien määrän funktiona, mutta kaiken kaikkiaan tämä ei muuta sitä, mikä on aikavaativuus piilokerroksen koon suhteen. Toki on syytä huomioida, että melko pienenkin verkon koulutus vie useita sekunteja per kierros, sillä dataa on todella paljon.

Työssä on runsaasti parannettavaa. Ihan perustavanlaatuinen parannus olisi se, että ohjelma voisi tukea sellaisia verkkoja, joissa on useampia piilokerroksia. Yritin pitkään implementoida tällaista toteutusta, mutta tämä johti syystä tai toisesta erinäisiin numeerisiin ongelmiin, minkä vuoksi päätin lopulta priorisoida yksinkertaisempaa, mutta toimivaa toteutusta. Joka tapauksessa yhdelläkin piilokerroksella verkon kyky luokitella testausdataan kuuluvia kuvia oli erittäin hyvä. Toteutin myös yksinkertaisen version niin sanotusta batchauksesta, jossa ideana on, että harjoitusdata jaetaan muutaman kuvan joukkoon, jotka käsitellään kerralla ja verkon parametreja päivitetään vasta koko batchin käsittelyn jälkeen siten, että gradientit keskiarvoistetaan. Toteutin tämän melko naiivisti laittamalla kaikki yksittäiset kuvat listaksi, mutta varmaan vähän paremmalla numpy-tuntemuksella ja enemmällä ajalla koodia olisi varmasti mahdollista vektoroida paremmin, jolloin suorituskyky saattaisi olla parempi.

Varsinaista ohjelmaa voisi parantaa myös. Yksi selkeä parannus liittyy siihen, että ohjelma on puhtaasti tekstipohjainen, mikä on tietysti vähän huono ratkaisu ottaen huomioon sen, että tarkoitus on tunnistaa kuvia. Näin ollen jonkinlainen graafinen käyttöliittymä tai edes mahdollisuus tallentaa kuvia tai esimerkiksi statistiikkaa olisi perusteltu. Mahdollisia muita mittareita verkkojen suorituskyvylle voisi olla esimerkiksi sekaannusmatriisin laskeminen.

En käyttänyt laajoja kielimalleja työssä.

Käytin varsinaisen kurssimateriaalin lisäksi seuraavia lähteitä:
https://tim.jyu.fi/view/143092#lis%C3%A4tietoa-aktivointifunktioista

https://www.sebastianbjorkqvist.com/blog/writing-automated-tests-for-neural-networks/
https://mattpetersen.github.io/softmax-with-cross-entropy