OULUN AMMATTIKORKEAKOULU

Roope Heinonen

HEDELMIEN JA VIHANNESTEN TUNNISTAMINEN NEUROVERKON AVULLA

HEDELMIEN JA VIHANNESTEN TUNNISTAMINEN NEUROVERKON AVULLA

Roope Heinonen
Opinnäytetyö
Kevät 2024
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä(t): Roope Heinonen
Opinnäytetyön nimi: Hedelmien ja vihannesten tunnistaminen neuroverkon avulla
Työn ohjaaja(t): Teemu Korpela
Työn valmistumislukukausi ja -vuosi: Kevät 2024 Sivumäärä: 33

Opinnäytetyön tavoitteena oli kehittää koneoppimismalli, joka kykenee tunnistamaan hedelmiä ja vihanneksia kuvasta, sekä luoda mobiilisovellus, jonka avulla käyttäjä voi seurata päivittäistä kokonaiskalorisaantiaan.

Raportissa perehdyttiin syvällisesti tekoälyyn ja sen eri alatekniikoihin, kuten syväoppimiseen ja neuroverkkoihin. Koneoppimismalli rakennettiin hyödyntämällä TensorFlow Lite ja Keras -ohjelmistokirjastoja, jotka mahdollistivat monikerroksisen konvoluutioverkon nopean kehittämisen.

Mobiilisovellus toteutettiin Javalla Androidille, ja se käyttää tätä kehitettyä koneoppimismallia hedelmien ja vihannesten tunnistamiseksi käyttäjän ottamista kuvista. Sovelluksessa on mahdollista tallentaa tunnistettujen kasvisten kaloriarvot päivittäiseen kokonaiskalorisaantiin. Lisäksi sovelluksella käyttäjä voi lisätä erilaisten ruokien ja juomien kalorimäärät viivakoodien skannaamisella.

Tuloksena syntyi toimiva Android-sovellus, joka yhdistää kehitetyn koneoppismallin ja kaloriseurannan. Tästä huolimatta mallin tunnistustarkkuus ja luotettavuus vaativat vielä parantamista.

Asiasanat: kuvantunnistus, koneoppiminen, neuroverkko, konvoluutioneuroverkko, tekoäly, mobiilisovellus, Android, Java

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

---

Author(s): Roope Heinonen
Title of thesis: Recognizing fruits and vegetables with neural network
Supervisor(s): Teemu Korpela
Term and year when the thesis was submitted: Spring 2024
Number of pages: e.g. 33

The goal of this thesis was to build a machine learning model, which can detect fruits and vegetables. In addition, another goal was to develop a mobile application that helps user to track their daily total caloric intake.

Thesis delved deeply into artificial intelligence and its sub technologies such as deep learning and neural networks. The machine learning model was built by using TensorFlow Lite and Keras software libraries which made developing a multilayer convolutional neural network fast.

The mobile application was developed with Java for Android, and it uses this built machine learning model to detect fruits and vegetables from user captured photos. In the application it's possible to save the calorific value from detected products to daily total caloric intake. In addition, in the application user may add calorific values from different foods and drinks by scanning the bar code.

The result was a working Android application that connects the developed machine learning model and calorie tracking. However, the recognition accuracy and reliability of the model still require improvement.

Keywords: image detection, machine learning, neural network, convolutional neural network, artificial intelligence, mobile application, Android, Java

SISÄLLYS

1 JOHDANTO

Ravitsemuksesta saa energiaa aineenvaihduntaan, fyysiseen aktiivisuuteen, ruuansulatuksen lämmöntuotantoon sekä muun muassa luurakenteiden muodostamiseen. Energiasaannin kasvaessa energiankulutusta suuremmaksi ylimääräinen energia varastoituu elimistöön rasvakudoksena sekä hiilihydraatteina. (1.) Ruokapäiväkirjan pitämisellä voidaan tarkastella kokonaisenergiansaantia.

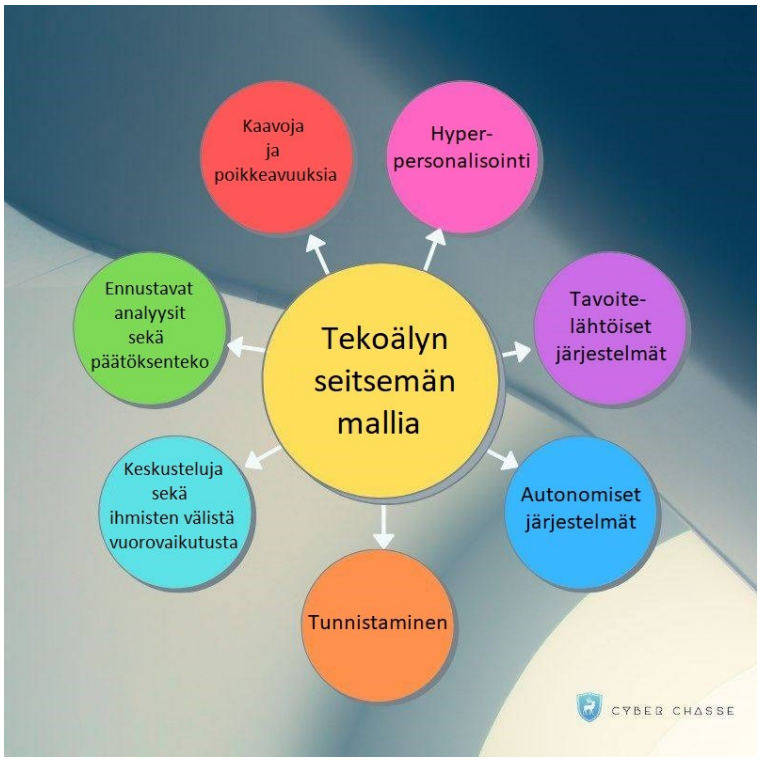
Tässä opinnäytetyössä kehitetään mobiilisovellus, joka toimii ruokapäiväkirjana ja hyödyntää tekoälyä. Raportissa keskitytään tekoälyn soveltamiseen hedelmien ja vihannesten tunnistamisessa. Tavoitteena on luoda toimiva ja tehokas koneoppimismalli, joka kykenee tunnistamaan erilaisia kasviksia kuvasta käyttäen syväoppimista ja konvoluutioneuroverkkoja. Tämän lisäksi työhön sisältyy mobiilisovelluksen kehitys, joka demonstroi luotua mallia ja mahdollistaa päivittäisen kalorisaannin seuraamista ruokapäiväkirjan muodossa. Sovelluksessa käyttäjän on mahdollista ottaa kameralla kuvia erilaisista kasviksista ja käyttää mallia tunnistamaan kyseiset kasvikset. Kun kuva on otettu, sovellus lisää kasviksen kalorimäärän päivittäiseen kaloriseurantaan.

Opinnäytetyössä käydään myös läpi tekoälyn ja syväoppimisen perusteita, koneoppimisen eri muotoja sekä neuroverkkojen ja konvoluutioneuroverkkojen toimintaperiaatteita. Näiden asioiden ymmärtäminen on tärkeää, jotta koneoppimismallin luominen kuvantunnistamiseen onnistuu.

2 TEKOÄLY, KONEOPPIMINEN JA NEUROVERKOT

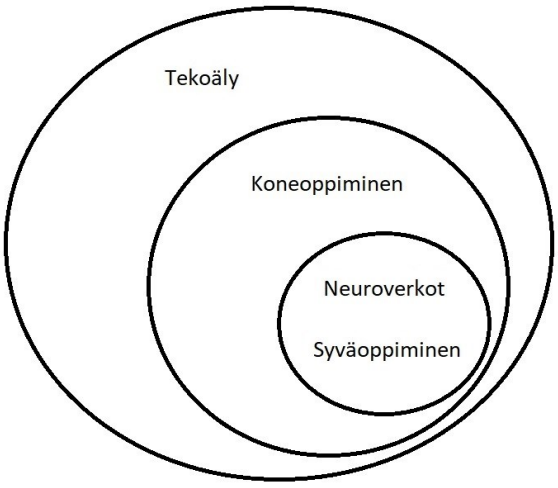
Tekoäly näkyy jokapäiväisessä elämässä, vaikka sitä ei tietoisesti käyttäisi. Sitä hyödynnetään tarjoamaan kohdennettuja mainoksia ja suosituksia ihmisille perustuen heidän ostos- ja hakuhistoriaansa. Kaupan alalla tekoälyä käytetään runsaasti tuotteiden parantamisessa, inventaarion rakentamisessa sekä logistiikassa. Verkon hakukoneissa tekoäly oppii jatkuvasti käsittelemään valtavaa määrää dataa tarjotakseen käyttäjille relevantteja hakutuloksia. Älypuhelinten digitaaliset avustajat osaavat vastata kysymyksiin, antaa suosituksia sekä auttaa ylläpitämään arjen rutiineja. Sarjat ja elokuvat hyödyntävät tekoälyä automaattisen tekstityksen sekä käännösten luomisessa. Älytermostaatit säästävät energiaa seuraamalla asiakkaan käyttäytymistä. Autot sisältävät nykyisin tekoälyyn perustuvia turvamekanismeja, jotka näkevät sekä havaitsevat vaaratilanteita. Tekoälyratkaisut parantavat kyberturvallisuutta analysoimalla datamäärää, tunnistamalla malleja sekä analysoimalla aiempia hyökkäyksiä ja vaaratilanteita. Maataloudessa tekoälyä voi käyttää vähentämään lannoitteiden, torjunta-aineiden sekä kastelun määrää ja lisäämään tuottavuutta sekä vähentämään viljelystä syntyviä ikäviä ympäristövaikutuksia. Tekoälyä voidaan lisäksi käyttää esimerkiksi nautakarjan ruoan kulutuksen sekä lääkityksen määrän valvomisessa. (2.) Maailmanlaajuisesti yrityksistä noin 35 % käyttää tekoälyä ja toiset 42 % tutkii teknologiaa (3).

Tekoälyjärjestelmä rakennetaan datan sekä algoritmien avulla. Kerättyä dataa sovelletaan matemaattisiin malleihin eli algoritmeihin, jotka tunnistavat datan sisältämiä rakenteita sekä mallintavat näitä rakenteita omassa päätöksenteossaan. Algoritmit hyödyntävät dataa tunnistaakseen kuvioita ja tehdäkseen ennusteita koulutusprosessin aikana, minkä jälkeen koulutetut algoritmit eri sovelluksissa voivat jatkuvasti oppia uudesta tiedosta ja sopeutua siihen. Tämän ansiosta tekoälyjärjestelmät voivat suorittaa monimutkaisia tehtäviä kuten kuvantunnistusta ja tietojen analysointia yhä tarkemmin ja tehokkaammin ajan myötä. (4.) Kuvassa 1 esitetään tekoälyn seitsemän eri mallia.



KUVA 1. Tekoälyn seitsemän mallia (muokattu lähteen 5 pohjalta)

Tekoälyjärjestelmän teknologioihin kuuluvat muun muassa koneoppiminen, syväoppiminen ja neuroverkot, jotka kaikki liittyvät toisiinsa (kuva 2). Koneoppiminen on tekoälyn osa-alue, syväoppiminen on koneoppimisen alaosio ja neuroverkot muodostavat syväoppimisalgoritmien perustan. (3.)



KUVA 2. Venn-diagrammi kuvaa tekoälyteknologioiden suhdetta toisiinsa

2.1 Koneoppimisen perusteet

Vuonna 1959 amerikkalainen Arthur Samuel keksi termin koneoppiminen, joka merkitsee erästä tekoälyn alaryhmää (6). Koneoppiminen keskittyy opettamaan koneita oppimaan datasta ja parantamaan kokemuksella, sen sijaan että ne olisivat ohjelmoituja siihen. Koneoppimisalgoritmeja opetetaan etsimään malleja sekä korrelaatioita suurista datamääristä ja tekemään ideaalisia päätöksiä sekä ennusteita analyysin perusteella. Näin saamansa datan avulla koneoppimismallit kehittävät itseään sekä muuttuvat sitä tarkemmaksi, mitä enemmän dataa niillä on käytettävissä. (7.) Datan määrä, laatu sekä rakenne ratkaisevat kuinka laadukas lopputulos on (8).

Koneoppiminen sisältää karkeasti kolme erityyppistä koneoppismallia, joissa käytetään erilaisia algoritmisia tekniikoita. Aineiston luonne ja haluttu lopputulos määräävät, valitaanko vaihtoehdoista ohjattu, ohjaamaton vai vahvistava malli. (7.)

Ohjattu oppiminen on ensimmäinen kolmesta mallista, jonka oppimisalgoritmeissa tietokonetta opetetaan määrätyillä säännöillä sekä esimerkkejä käyttäen. Mallit koostuvat ”tulo”- ja ”tuotos” -tietopareista, joista jälkimmäiseen on merkitty haluttu arvo. Kuvitellaan esimerkkinä tilanne, jossa tavoitteena on saada tietokone kertomaan ero banaanin ja omenan välillä. Yksi binäärinen syöttötietopari sisältää kuvan banaanista sekä omenasta. Haluttu tulos kyseisestä parista on valita banaani, jolloin se määritetään etukäteen oikeaksi vastaukseksi. Kerätessään harjoitusdataa järjestelmä alkaa algoritmin avulla määrittämään yhtäläisyyksiä ja eroavaisuuksia materiaaleissa, kunnes se pystyy ennustamaan vastauksen täysin itsenäisesti. (7.)

Toisin kuin ohjatussa oppimisessa, ohjaamattomassa oppimisessa ei anneta valmista ratkaisua. Kone analysoi dataa, jossa suuri osa materiaalista on merkitsemätöntä sekä jäsentymätöntä, ja alkaa tunnistaa kuvioita ja rakenteita materiaalissa käyttäen saatavilla olevaa tietoa. Malli kuvaa tapaa, jolla ihmiset tarkkailevat ja ryhmittelevät asioita: mitä enemmän esimerkkejä kerääntyy, sitä enemmän kokemusta saadaan ryhmitellä ja tunnistaa asioita entistä tarkemmin. Markkinatutkimuksessa, kyberturvallisuudessa ja kasvojen tunnistamisessa käytetään tätä koneoppimismallia. (7.)

2.2 Syväoppimisen toimintaperiaate

Syväoppiminen on eräs koneoppimisen alaryhmä, joka hyödyntää syviä neuroverkkoja ihmismäisen päätöksentekokyvyn jäljittelemiseksi. Neuroverkot ovat ihmisen aivosolujen toimintaan pohjautuva koneoppimisen menetelmä. Suurimmassa osassa tekoälysovelluksista on nykyisin käytetty eri syväoppimismenetelmiä. (9.)

Rakenne neuroverkkoarkkitehtuurissa muodostaa merkittävän eron koneoppimisen ja syväoppimisen välillä. Perinteisissä koneoppimismalleissa käytetään yksinkertaisia neuroverkkoja, joissa on yksi tai enintään kaksi kerrosta. Sen sijaan syväoppimismalleissa kerroksia on vähintään kolme, ja niiden määrä voi nousta jopa useisiin tuhansiin. (9.)

Valvotun koneoppismallin vaatiessa jäsenneltyä ja merkittyä dataa tarkkojen tuloksen saavuttamiseksi, syväoppismallit voivat käyttää valvomatonta oppimismallia. Tämän antaa syväoppimismalleille mahdollisuuden tunnistaa ja eritellä olennaisia piirteitä raa'asta datasta, joka saattaa olla jäsentelemätöntä. Lisäksi syväoppimismallit kykenevät arvioimaan sekä tarkentamaan tuloksiaan parantaakseen tarkkuuttaan. (9.)

Neuroverkot yrittävät mallintaa ihmisen aivojen toimintaa yhdistämällä dataa, painotuksia ja rajaarvoja silikonineuroneissa. Nämä neuronit ovat toisiinsa kytkettyjä useilla eri kerroksilla, joissa jokainen neuroni tarkentaa edellisen kerroksen dataa. Mallin näkyväksi kerrokseksi nimitetään syöttö- ja lähtökerrosta. Ensimmäisessä syöttökerroksessa syväoppimismalli vastaanottaa datan prosessointia varten, kun taas viimeisessä lähtökerroksessa tehdään ennuste tai luokittelu. Syväoppiminen tarvitsee runsaan määrän laskentatehoa, jolloin korkean suorituskyvyn graafiset prosessointiyksiköt ovat ideaalisia ratkaisuja suuremman muistikapasiteettinsa ansiosta. (9.)

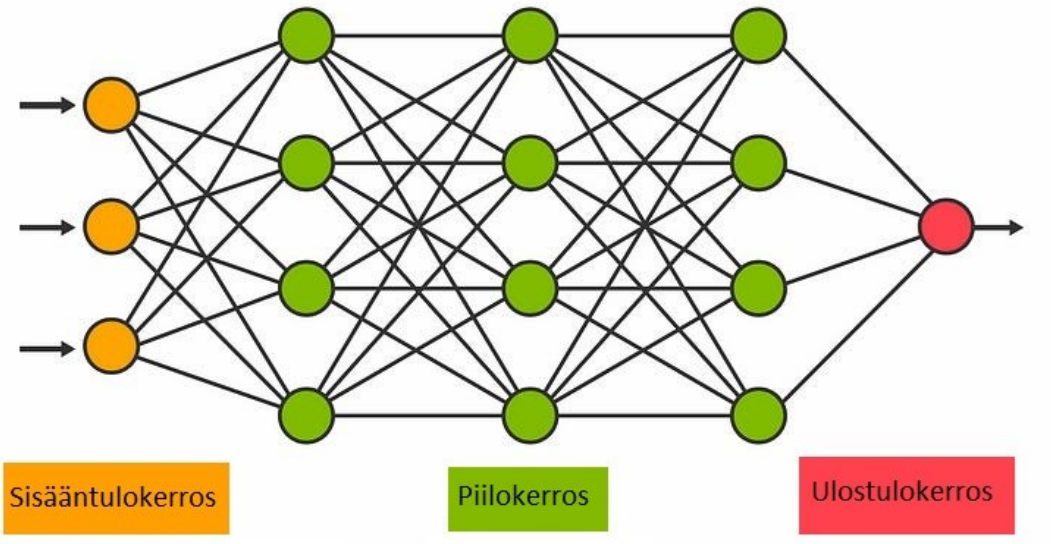
Syväoppimisalgoritmit ovat erittäin monimutkaisia ja erilaisia tarkoituksia varten on erilaisia neuroverkkoja. Tärkeimpiä algoritmeja on muun muassa konvoluutioneuroverkko, toistuva neuroverkko, autoenkooderi ja muuttuva autoenkooderi, diffuusiomalli sekä transformermalli. Nämä menetelmät ovat parantaneet merkittävästi kuvan- ja puheentunnistuksen sekä lukuisten eri alojen, kuten lääkekehityksen, viimeisintä tekniikkaa. (9.)

2.3 Neuroverkkojen rakenteet ja algoritmit

Neuroverkko on yksinkertaisuudessaan koneoppimismalli, joka matkii ihmisaivojen toimintaa. Se käyttää prosesseja, jotka jäljittelevät biologisten hermosolujen yhteistyötä ilmiöiden tunnistamisessa, vaihtoehtojen punnitsemisessa ja päätösten tekemisessä. Neuroverkkojen historia ulottuu vuoteen 1943, jolloin Warren S. McCulloch ja Walter Pitts julkaisivat tutkimuksen siitä, kuinka ihmisen aivot pystyvät tuottamaan monimutkaisia malleja toisiinsa liittyvien hermosolujen avulla. Yksinkertaisten neuroverkkojen rakentamista yritettiin jo 1950-luvulla, mutta niiden suosio kasvoi merkittävästi vasta 1980-luvulla. Vuonna 1989 neuroverkko onnistuneesti tunnisti Yhdysvalloissa käsin kirjoitetut postinumerot kirjeestä. (10.)

Neuroverkot koostuvat prosessointiyksiköistä, joita kutsutaan neuroneiksi. Nämä neuronit kommunikoivat keskenään ja välittävät tietoa toisilleen, aivan kuten aivoissa hermosolut välittävät sähköimpulsseja toisilleen. Neuroverkkoja käytetään koneoppimisessa ja erityisesti syväoppimisessa. Syväoppimismalli, joka perustuu neuroverkkoon ja on opetettu riittävällä datamäärällä, kykenee tunnistamaan esimerkiksi kohteita kuvasta, jota se ei ole koskaan ennen nähnyt. (11.)

Neuroverkon neuronit on levitetty vähintään kolmelle kerrokselle, jotka ovat sisääntulokerros, piilokerros ja ulostulokerros. Neuroverkoissa voi olla useita piilokerroksia, mikä tekee verkosta syvän (kuva 3). Mitä syvempi verkko on, sitä enemmän se vaatii prosessointitehoa, mutta samalla se kykenee suorittamaan monimutkaisempia tehtäviä. (11.)



KUVA 3. Syvän neuroverkon arkkitehtuuri (muokattu lähteen 12 pohjalta)

Jokaisessa kerroksessa neuroni suorittaa prosessointitehtävän tai funktion edellisen kerroksen lähettämällä syötteellä. Neuronit käyttävät matemaattista kaavaa, jossa jokainen muuttuja on painotettu eri tavalla. Jos kaavan soveltamisen tulos syötteelle ylittää tietyn rajan, neuroni välittää datan seuraavalle kerrokselle. Jos tulos on vastaavasti rajan alapuolella, dataa ei välitetä eteenpäin. (11.)

Jokainen yksittäinen neuroni voidaan käsittää omana lineaarisena regressiomallinaan, johon sisältyy siis syöte, painoarvo, raja-arvo ja tulos (10). Kaava 1 havainnollistaa kuinka neuronin painettu summa lasketaan.

KAAVA 1. Neuronin matemaattinen kaava (10).

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

$$y = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

Kaavassa 1 jokainen syöte *x* kerrotaan vastaavilla painoilla *w* ja summataan yhteen. Lopuksi summaan lisätään raja-arvo *b*.

Esimerkki painetun summan laskemisesta kaavalla 1 jos syötteiden arvot ovat $(x_1 = 3, x_2 = 0.8, x_3 = 7)$, painoarvot ovat $(w_1 = 2, w_2 = 0.7, w_3 = -0.5)$ ja raja-arvo on $(b = 0)$:

$$y = (2 * 3) + (0.7 * 0.8) + (-0.5 * 7) + 0$$

$$y = 3.06$$

Raja-arvolla tulosta saisi esimerkiksi kasvatettua, jos tulos ei muuten riitä ylittämään aktivaatiofunktion rajaa ja halutaan varmistaa, että neuroni aktivoituu tietyissä tilanteissa.

Painotettu summa siirretään seuraavaksi aktivaatiofunktion käsiteltäväksi. Aktivaatiofunktioiden päätyyppejä on binääriaskelfunktio, lineaarinen aktivaatiofunktio sekä epälineaariset aktivaatiofunktiot. Binääriaskelfunktio perustuu raja-arvoon, joka määrittää aktivoidaanko neuroni vai ei. Jos painotettu summa on sitä suurempi, neuroni aktivoituu ja lähettää ulostulon 0 tai 1 seuraavalle kerrokselle neuroverkossa. Lineaarisessa aktivaatiofunktiossa ei ole mitään raja-arvoa vaan aktivointi on suoraan verrannollinen syötteeseen eikä funktio suorita mitään laskentatoimenpiteitä painotetulle summalle. (13.)

Epälineaarisia aktivaatiofunktioita on useampia. Ne mahdollistavat useiden kerrosten pinoamisen, sillä syöte on nyt useiden kerrosten läpi kulkevien tulojen epälineaarinen yhdistelmä. Sigmoid-funktiossa ulostulo on arvojen 0 ja 1 välillä. Suuremmilla syötteillä ulostulo lähenee arvoa 1.0, kun taas

pienemmillä syötteillä ulostulo on lähempänä arvoa 0 (kaava 2). Sigmoid-funktio on yksi yleisimmin käytetyistä aktivaatiofunktioista ja sitä käytetään yleisesti malleissa, joissa joudutaan ennustamaan todennäköisyyksiä. Funktio on myös differentioituva ja tarjoaa tasaisen gradientin, mikä muun muassa estää ulostuloarvojen äkilliset vaihtelut. (13.)

KAAVA 2. Sigmoid-funktion kaava (13).

$$f(x) = \frac{1}{1 + e^{-x}}$$

Kaavassa 2 *x* on syöte.

Toinen suosituimmista aktivaatiofunktioista on ReLU-funktio, tai Rectified Linear Unit. Jos syöte on 0 tai vähemmän, ulostulon arvo on 0. Kun syötteen arvo ylittää nollan, ulostulo vastaa syötteen arvoa kuten kaavasta 3 voidaan nähdä. (13.)

KAAVA 3. ReLU-funktion kaava (13).

$$f(x) = \max(0, x)$$

Kaavassa 3 *x* on syöte.

Koska ReLU-funktio aktivoi vain tietyn määrän neuroneja kerrallaan, funktio on laskennallisesti paljon tehokkaampi kuin Sigmoid-funktio. Funktion heikkoutena on, että se nollaa negatiiviset syötteet, mikä heikentää mallin tarkkuutta opettamistilanteessa. (13.)

Leaky ReLU -funktio on paranneltu versio alkuperäisestä ReLU-funktiosta. Jos nyt syötteen arvo on negatiivinen, palauttaa funktio pienen negatiivisen arvon, joka on suoraan verrannollinen syötteeseen kuten kaava 4 osoittaa. Funktion huonona puolena on pidentynyt mallin opettamiseen vaadittava aika. (13.)

KAAVA 4. Leaky ReLU -funktion kaava (13).

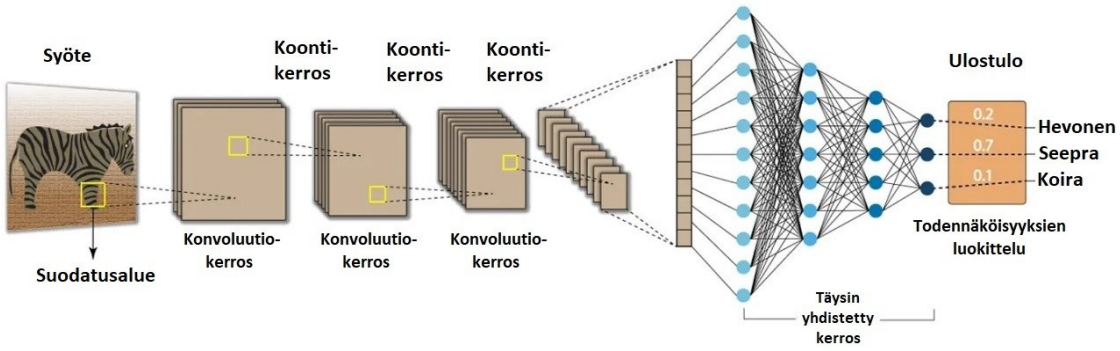
$$f(x) = \max(0.1x, x)$$

Kaavassa 4 *x* on syöte.

2.4 Konvoluutioneuroverkkojen rooli kuvantunnistuksessa

Konvoluutioneuroverkot eroavat muista neuroverkoista poikkeuksellisen suorituskykynsä ansioista erityisesti kuvien, puheen ja äänisignaalien käsittelyssä. Ne koostuvat kolmesta erilaisesta kerrostyypistä, joita ovat konvoluutiokerros, koontikerros (engl. pooling layer) ja täysin yhdistetty kerros (engl. fully connected layer). Ensimmäisenä kerroksena on konvoluutiokerros, ja tällaisia kerroksia

voi olla useita peräkkäin. Yleensä jokaisen yksittäisen konvoluutiokerroksen jälkeen seuraa koontikerros, ja verkon viimeinen kerros on täysin yhdistetty kerros. Jokaisen kerroksen myötä verkon monimutkaisuus kasvaa, sillä se kykenee tunnistamaan yhä suurempia ja monimutkaisempia osia kuvasta. (14.) Kuva 4 havainnollistaa verkon toimintaperiaatetta.

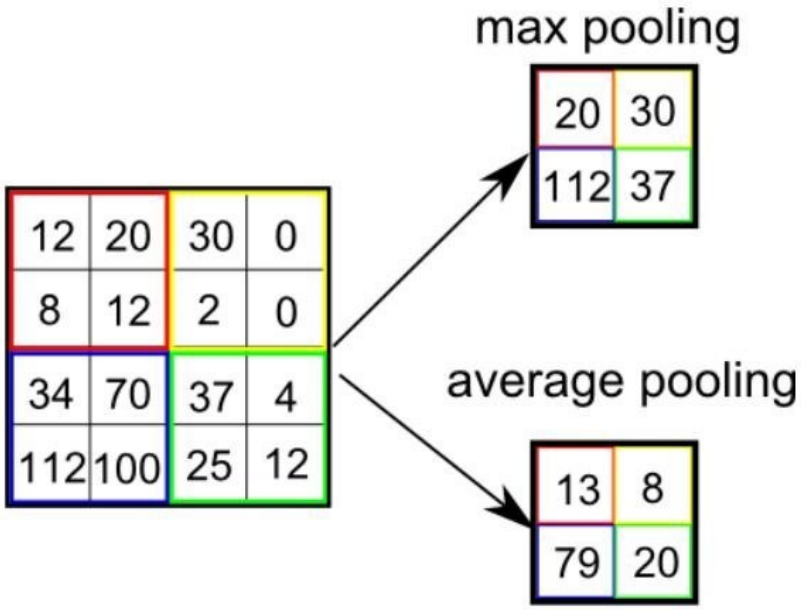


KUVA 4. Konvoluutioneuroverkon toimintaperiaate (muokattu lähteen 15 pohjalta)

Konvoluutiokerros on konvoluutioverkon ydinosa, jossa suurin osa laskennallisesta työstä tapahtuu. Jos syötteenä on värillinen kuva, siinä on kolme ulottuvuutta: korkeus, leveys ja syvyys, joka vastaa kuvissa värikanavien määrää. Kerros koostuu yhdestä tai useammasta kaksiulotteisesta suodattimesta, jotka kattavat vain pienen osan kuvasta, ja ne etsivät kuvasta erilaisia ominaisuuksia. Kun suodatin lisätään kuvan alueelle, sen ja syöttöpikseleiden välille lasketaan pistetulo, eli tehdään konvoluutio-operaatio. Tämän jälkeen suodatin siirtyy seuraavalle alueelle, kunnes se on käsitellyt koko kuvan. Suodattimesta saatu pistetulojen sarja tuottaa aktivointikartan, johon sovelletaan ennalta määritettyä aktivaatiofunktiota. (14.)

Koontikerroksia lisätään yleensä jokaisen konvoluutiokerroksen jälkeen. Yleisin muoto koontikerroksen suodatukselle on 2 x 2 pikseliä, jota käytetään kahden pikselin askeleella. Tämä vähentää jokaisen aktivointikartan ulottuvuuksia puoleen, mikä tarkoittaa, että kartan pikselien määrä pienenee neljänneksen. Esimerkiksi 6 x 6 pikselin aktivointikarttaan lisätty koontikerros pienentää kartan 3 x 3 pikselin kokoiseksi. Toisin sanoen 36 pikseliä kutistuu 9 pikseliin. (16.)

Koontikerrosten kaksi yleisintä yhdistämisfunktiota ovat average pooling -funktio, jossa lasketaan jokaisen kohdealueen keskiarvo, ja max pooling -funktio, jossa lasketaan kohdealueen enimmäisarvo. Max pooling -funktion on todettu toimivan käytännössä paremmin konenäkötehtävissä, kuten kuvan tunnistamisessa. (16.) Funktioiden toimintaperiaatteet ovat esitetty kuvassa 5.



KUVA 5. Koontikerroksen yhdistämisfunktiot (17)

Viimeisessä täysin yhdistetyssä kerroksessa jokainen neuroni ulostulokerroksessa on suoraan yhteydessä edellisen kerroksen neuroniin. Tämä kerros luokittelee kuvan sen perusteella, mitä ominaisuuksia on poimittu edellisten kerrosten ja niiden suodattimien avulla. (14.)

3 KONEOPPIMISMALLIN LUOMINEN

Koneoppimismallin kehitysympäristöksi valitsin Googlen tarjoaman Colaboratoryn, lyhenteeltään Colab. Vaihtoehtoisia palveluita oli muun muassa Jupyter Notebook tai Pythonin jakelupaketti Anaconda, mutta päädyin Colabiin lähinnä sen helppokäyttöisyyden takia. Ohjelmointikieleksi valitsin Pythonin, sillä minulla on siitä jonkin verran aikaisempaa kokemusta ja se on yksi suosituimpia kieliä koneoppimisen alalla. Lisäksi Python on suosituin kieli TensorFlow-ohjelmakirjaston käyttämiseen (18).

Colaboratory on pilvipalvelu, joten koneoppimismallin opettaminen onnistuu pelkästään selaimella ilman ylimääräisten sovellusten asentamista ja ohjelmakoodi Colabissa suoritetaan Googlen pilviresursseilla riippumatta oman laitteen tehoista. Colab tarjoaa myös ilmaisen grafiikkasuoritinkiihdytyksen, mikä voi huomattavasti nopeuttaa algoritmien opettamista. (19.)

TensorFlow on Googlen julkaisema avoimen lähdekoodin ohjelmakirjasto, joka tukee monipuolisesti laskentaa vaativia tehtäviä, kuten syväoppimista, koneoppimista ja neuroverkkoja. Kirjasto on kirjoitettu pääasiassa C++-kielellä, mutta Pythonia käytetään yhdistämään eri osat korkean tason ohjelmointiabstraktioiden avulla. TensorFlow pohjautuu niin kutsuttuihin tensoreihin, jotka ovat moniulotteisia vektoreita ja matriiseja matemaattisten operaatioiden välillä. Kirjasto luo laskennallisen graafin, joka ohjaa datan virtaamista ja määrittää mallin opettamista. Sitä voidaan käyttää syvien neuroverkkojen opettamiseen ja hyödyntää esimerkiksi kuvantunnistuksessa, käsinkirjoituksen tunnistuksessa tai luonnollisen kielen käsittelyssä. (18.) Twitter on myös käyttänyt kirjastoa twiittien priorisoimiseen, jotta käyttäjät näkisivät tärkeimmät twiitit, vaikka he seuraisivat tuhansia käyttäjiä (20).

TensorFlow-sovellukset voidaan opettaa hyödyntämällä joko laitteen prosessoria tai näytönohjaimen grafiikkasuoritinta. Grafiikkasuorittimella opettaminen on merkittävästi nopeampaa, koska näytönohjaimen tuhannet pienet ja tehokkaat ytimet voivat käynnistää tuhansia rinnakkaisia säikeitä samanaikaisesti, mikä on erityisen hyödyllistä laskentaintensiivisissä tehtävissä kuten matriisioperaatioissa ja algebrallisissa laskelmissa, joista molemmista syväoppiminen on vahvasti riippuvainen. Valmiit opetetut TensorFlow-mallit toimivat lukuisilla eri alustoilla aina mobiililaitteista huippuluokan palvelimiin. (20.)

Keras on korkeantason ohjelmistorajapinta syväoppimiselle, ja se on suunniteltu erityisesti helpottamaan ja nopeuttamaan neuroverkkojen rakentamista. Se tarjoaa helppokäyttöisen ja modulaarisen rajapinnan muun muassa TensorFlow-kirjastolle, ja se on myös yksi käytetyimmistä kirjastoista neuroverkkojen kehittämiseen. Kirjastolla verkon kerrosten ja monimutkaisten arkkitehtuurien luominen on todella yksinkertaista. Lisäksi Keraksen malleille on olemassa täysin konfiguroitavia moduuleja, kuten optimointifunktioita, aktivaatiofunktioita sekä alustusmalleja. Yhtenä etuna on myös se, että uusia ominaisuuksia voidaan helposti lisätä erillisinä moduuleina. (21.)

3.1 Mallin valmisteluvaihe ja aineiston esikäsittely

Mallin opettamiseen päätin käyttää valmista ja laadukkaalta vaikuttavaa kuvatietojoukkoa (engl. image dataset), joka sisältää kuvia kymmenestä eri hedelmästä ja 26:sta eri vihanneksesta. Jokaisesta lajista on 100 kuvaa opetustietojoukossa, kymmenen kuvaa validointitietojoukossa ja myös kymmenen kuvaa testitietojoukossa. Kuvatietojoukon tekijä on kertonut, että kuvat kasviksista on kerätty hänen omaan projektiinsa käyttämällä Bingin kuvahakua (22).

Loin testimallin käyttämällä tätä kuvatietojoukkoa ja huomasin, että kuvien joukossa oli paljon epäolennaisia kuvia. Esimerkiksi oli sumeita kuvia, kuvia, joissa itse kasvis oli vain osittain näkyvissä, ja jopa ihan piirrettyjä kuvia. Enemmistö kuvista oli kuitenkin ihan kelvollisia, joten päätin jatkaa tällä datasetillä. Testasin mallia lataamalla muutaman hedelmän kuvan Googlen kuvahaulla, mutta malli näytti jopa ihan selville hedelmän kuville vääriä vastauksia. Tulin siihen tulokseen, että opetuskuvien vähäisen määrän sekä joukossa olevien epäolennaisten kuvien takia poistan kuvatietojoukosta harvinaisempia ja liian samankaltaisia lajeja. Mikäli olisin halunnut käyttää kaikkia kasviksia, vaatisi se laadukkaamman opetuskuvajoukon. Käytännössä tämä tarkoittaisi, että jokainen kuva kasviksesta ei sisältäisi mitään ylimääräistä, kuvassa näkyisi selvästi kyseinen kasvis ja kuvia olisi satoja jokaisesta lajista eri kuvakulmasta.

Kasvislajien vähentämisen jälkeen testimalli antoi edelleen hälyttävän paljon vääriä vastauksia selvistä kuvista, joten päätin yhdistää kaksi kuvatietojoukkoa yhdeksi. Tässä toisessa datasetissä jokainen hedelmä ja vihannes on kuvattu asettamalla ne hitaasti pyörivän moottorin akselin päälle, jonka jälkeen kasviksesta on kuvattu lyhyt 20 sekunnin video valkoista taustaa vasten (23). Lopuksi videosta on taltioitu kuvasarja (engl. image sequence), jolloin jokaisesta kasviksesta on saatu keskimäärin 470 kuvaa. Olisin voinut käyttää pelkästään tätä jälkimmäistä datasettiä, mutta halusin

että kaikki kuvat eivät olisi samalla taustalla. Muuten olisin todennäköisesti joutunut kuvaamaan testikuvat myös valkoista taustaa vasten.

Viimeisenä latasin valmiin pakatun kuvatietojoukon omalle Google Drive -tililleni, josta se oli helposti ja nopeasti siirrettävissä Colaboratoryn työtilaan kuten kuva 6 osoittaa.

```
from google.colab import drive
drive.mount("/content/drive")
!cp "/content/drive/MyDrive/fruits_dataset.zip" fruits_dataset.zip
drive.flush_and_unmount()
!unzip fruits_dataset.zip
```

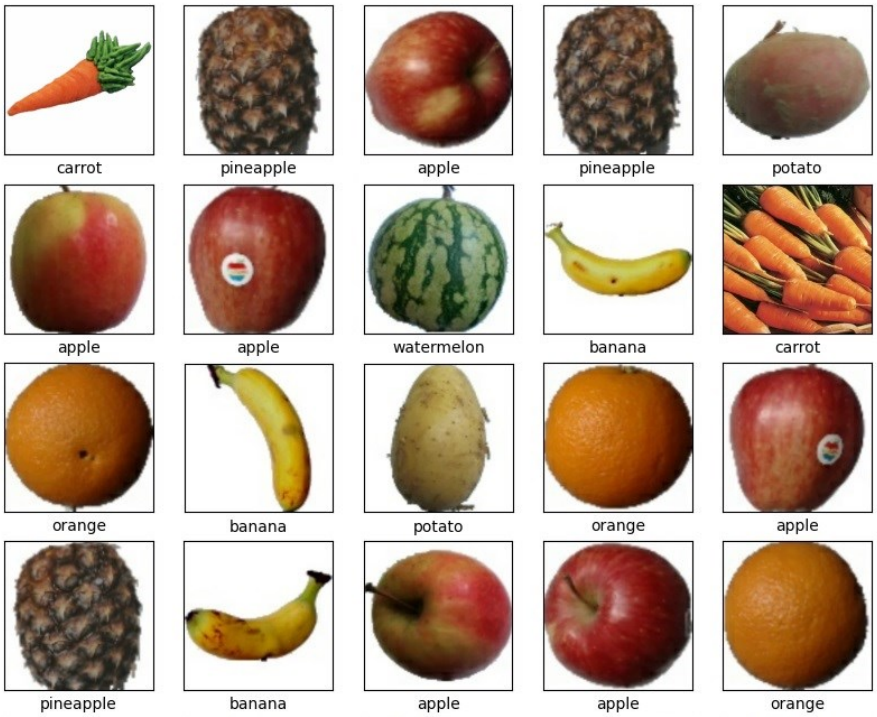
KUVA 6. Kuvatietojoukon lataaminen Colaboratoryn työtilaan

3.2 Opettamisprosessi ja mallin optimointi

Varsinaista mallia tehdessäni määrittelin käsiteltävän kuvan kooksi 256 pikseliä korkeussuunnassa ja 256 pikseliä pituussuunnassa. Huomasin, että liian tarkoissa kuvissa malli voi ottaa häiriöitä kuvien taustasta, mutta myös opettaminen pienemmillä kuvilla on nopeampaa.

Opetus-, validointi- ja testaustietojoukot loin käyttämällä Kerasin tarjoamaa image_dataset_from_directory -funktiota. Funktion parametreihin määrittelin kuvien sijainnin lisäksi kuvien sattumanvaraisen sekoituksen, opetettavan kuvajoukon koon ja opetuskierrosten määrän. Opetusvaiheessa ohjelmakoodi pilkkoo opetettavat kuvat erillisiksi kuvajoukoiksi, jonka jälkeen malli kouluttaa itseään toistuvasti iteroimalla koko kuvatietojoukon läpi opetuskierrosten määrän verran (24).

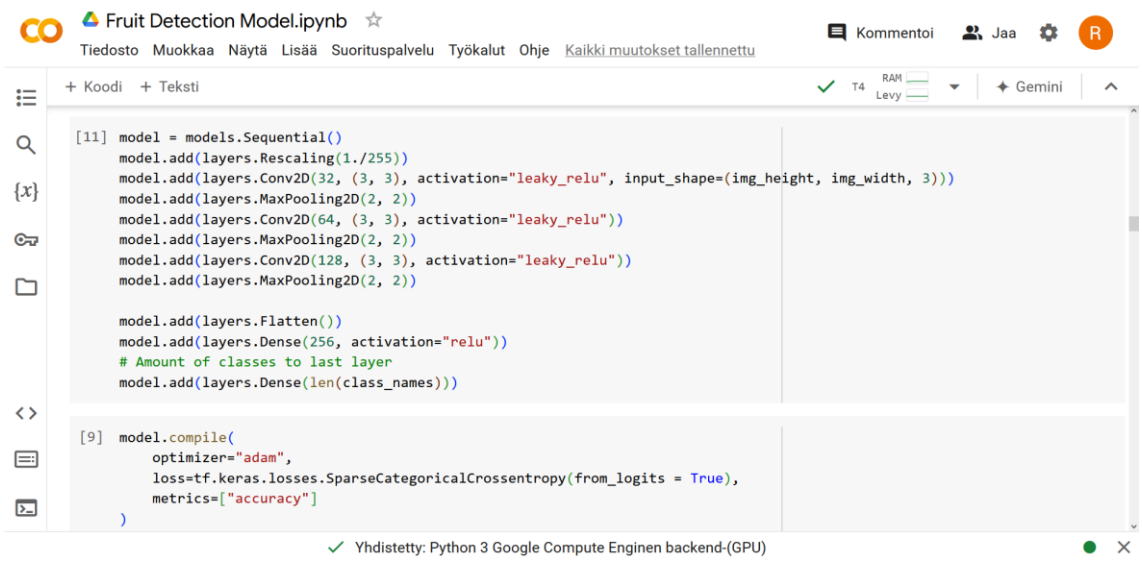
Käyttämällä Matplotlib-kirjastoa pystyin visualisoimaan ja varmistamaan, että opetustietojoukon kuvat ja luokat on luettu oikein. Kuvassa 7 on visualisoitu sattumanvaraisesti 20 kasvista luokkineen. Kasvisten luokkien nimet sain opetustietojoukosta.



KUVA 7. Opetuskuvien visualisointi

Seuraavaksi loin Kerasin Sequential-mallille kerrokset. Ensimmäinen kerros on esikäsittelykerros, joka käsittelee sisään tulevan kuvan skaalaamalla kuvan jokaista pikseliarvoa. Pikseliarvot ovat normaalisti välillä 0–255, mutta skaalaamalla ne välille 0–1 opettaminen helpottuu huomattavasti, koska mallin ei tarvitse käsitellä niin erilaisia arvoja suurella vaihteluvälillä.

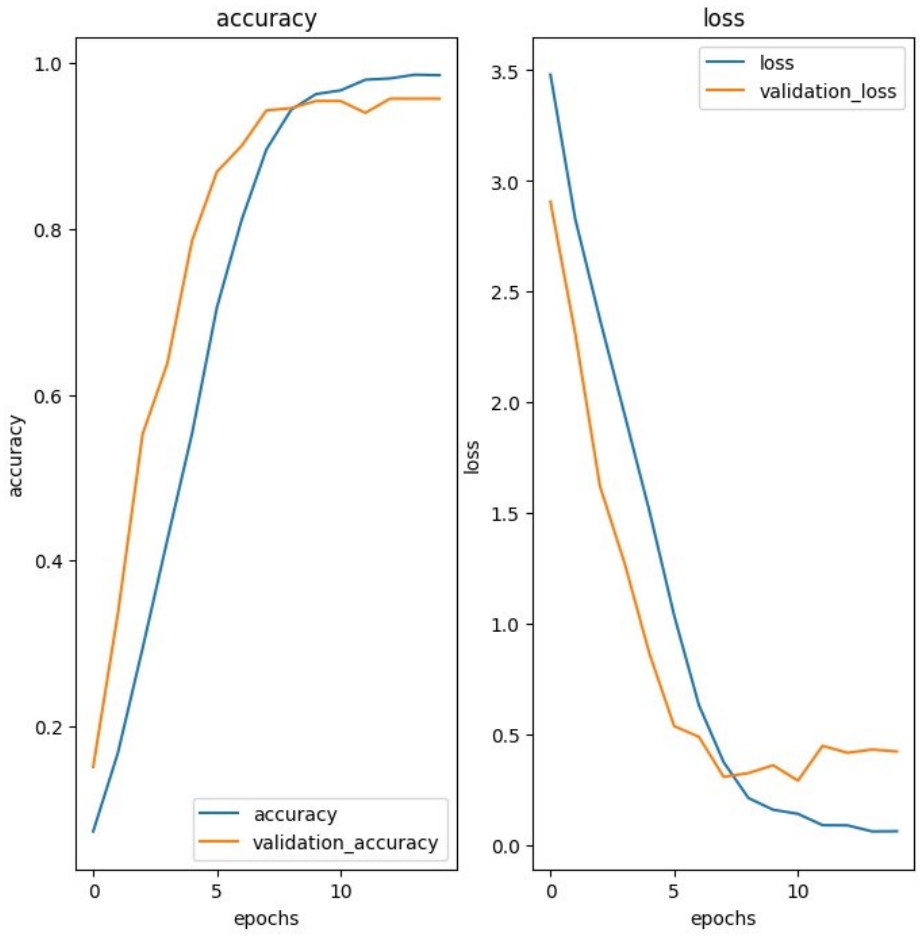
Tämän jälkeen malli sisältää kolme kaksiulotteista konvoluutiokerrosta kuten kuva 8 osoittaa. Ensimmäiseen konvoluutiokerrokseen määritetään sisään tulevan datan muoto, joka on kuvan koko pikseleissä ja kuvan värikanavien määrä. Käytin kerroksissa Leaky ReLU -aktivaatiofunktiota, jolla sain luotettavampia tuloksia kasvisten tunnistamisessa kuin perinteisellä ReLU-funktiolla, koska Leaky ReLU -funktio ei nollaa negatiivisia arvoja. Jokaisen konvoluutiokerroksen perään laitoin koontikerroksen 2 x 2 pikselillä, joka pienentää edellisen konvoluutiokerroksen pikselimäärää neljänneksellä. Koontikerrokset käyttävät max pooling -koontifunktiota. Täysin yhdistetyssä kerroksessa käytin tavallista ReLU-aktivaatiofunktiota. Vaihtoehtoisesti olisin voinut käyttää Sigmoid-funktiota tai siihen rinnastuvaa Softmax-funktiota, mutta en huomannut tuloksessa merkittävää eroa ReLU-funktioon.



KUVA 8. Mallin neuroverkon kerrokset

Mallin kääntämiseen käytin Adam-optimointifunktiota, jonka etuina on muun muassa dynaaminen oppimisnopeus jokaiselle mallin parametrille optimaalisen oppimisen saavuttamiseksi sekä sen tehokkuus käsitellä harvassa olevaa dataa. Kääntäjän häviöfunktiona toimi erityisesti luokitteluun soveltuva sekä neuroverkkojen opettamiseen optimisoitu SparseCategorialCrossentropy-funktio. (25.) Opetuskertojen määräksi valitsin 15.

Kääntämisen jälkeen loin kuvaajan mallin opetustietojoukon sekä validointitietojoukon tarkkuusarvosta ja häviöarvosta suhteutettuna opetuskertojen määrään. Kuten kuvasta 9 voidaan nähdä, opetustietojoukon tarkkuusarvo ylitti 90 % jo yhdeksännen opetuskierroksen jälkeen ja häviöarvo tippui alle 0.1:en yhdennellätoista kierroksella, jonka jälkeen molemmat arvot tasoittuivat. Validointitietojoukon kuvien vähäisen määrän takia sen tarkkuus sekä häviöarvot heittelivät rajusti vielä viimeisillä kierroksilla eivätkä lopulta yltäneet opetustietojoukon arvojen tasolle. Kääntämiseen kului noin 20 minuuttia Colaboratoryn grafiikkasuoritinkiihdytyksellä. Ajoittain grafiikkasuoritinkiihdytys ei ollut saatavilla sivuston suuren kävijämäärän takia, mikä johti siihen, että pelkällä prosessorilla kääntäminen vei lähes neljä tuntia.

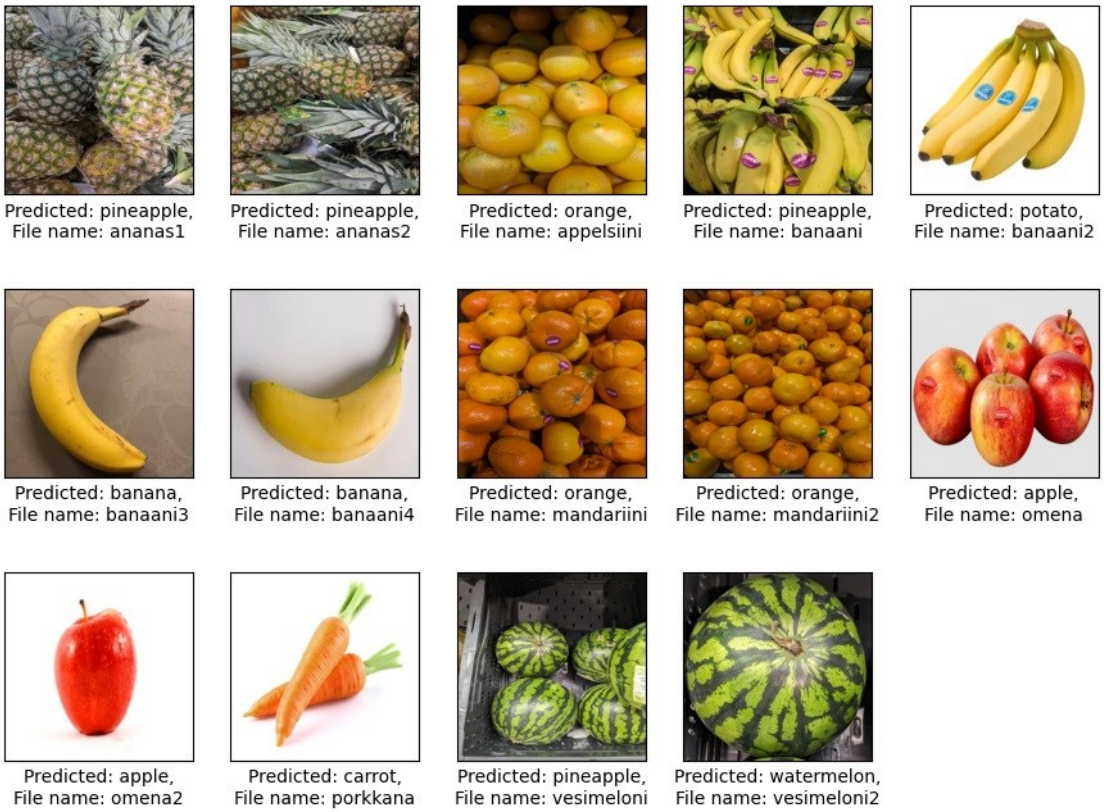


KUVA 9. Kuvaaja mallin tarkkuusarvosta ja häviöarvosta

3.3 Testaaminen ja suorituskyvyn arviointi

Mallin testaaminen testaustietojoukolla tapahtui helposti yhdellä komennolla. Evaluate-komento antoi testaustietojoukon tarkkuudeksi lopulta jopa 96 %. Jälkimmäisen kuvatietojoukon testauskuvat olivat osittain samoja tai hyvin samankaltaisia kuin opetustietojoukossa, joten mallin oli helppo tunnistaa kasvikset kyseisistä kuvista, mikä ehkä hieman vääristi todellista tarkkuusprosenttia.

Päätin lopuksi vielä testata mallia itse ottamillani kuvilla sekä suomalaisilta sivustoilta löytyvillä kuvilla, joita malli ei ole voinut nähdä aikaisemmin. Omat kuvani olin ottanut puhelimellani joko kaupan hedelmäosastolla tai kotonani. Sijoitin kuvat taulukkoon ja kirjoitin jokaisen kuvan kohdalle sen tiedostonimen ilman päätettä sekä mallin tunnistuksen tuloksen (kuva 10).



KUVA 10. Testattavien kuvien visualisointi

Tuloksista voidaan tehdä useita johtopäätöksiä. Lähtökohtaisesti on selvää, että malli osasi hyvin tunnistaa oikeat lajit kuvasta. On kuitenkin huomattava, että mallilla oli vaikeuksia tunnistaa oikeaa lajia silloin, kun kuvassa oli useampia samanlaisia hedelmiä. Tämä johtuu siitä, että opetustietojoukon kuvissa jokainen kuva sisältää vain yhden kappaleen kyseistä lajia, jolloin malli oppi tunnistamaan lajeja vain yksittäisten esimerkkien perusteella. Kun kuvassa on useampi samankaltainen hedelmä, mallin kyky erotella yksilöiden välisiä eroja heikkenee. Laajemmalla ja monipuolisemmalla kuvatietojoukolla mallin tunnistustarkkuus paranisi vastaavissa tilanteissa.

4 DEMOSOVELLUS

Koneoppimismallin testaamiseksi päätin luoda Javalla Android-sovelluksen, joka mahdollistaa kalorien seurannan. Sovellus tunnistaa käyttäjän lähettämistä kuvista kasviksia ja laskee niiden kalorimäärän osaksi päivän kokonaiskalorisaantia. Sovelluksella on myös mahdollista lukea tuotteiden viivakoodeja ja hakea niiden kalorimäärät Open Food Facts -sivustolta. Ulkoasun jätin aika yksinkertaiseksi, koska tämän sovelluksen tarkoitus oli vain testata kuinka malli toimisi Android-sovelluksessa.

4.1 Koneoppimismallin integrointi Android-sovellukseen

Koneoppismallin tuonti Android Studion projektiin on hyvin yksinkertaista. Tuonnin yhteydessä ohjelmointiympäristö tarjoaa jopa esimerkkikoodin, kuinka mallilla voidaan tunnistaa kuvia. Koodi ei kuitenkaan sisällä kuvan esikäsittelyä tai pikselien muuntamista mallille sopivaan muotoon.

Tunnistusta varten kuvasta luodaan ensin Bitmap-olio, jonka koko skaalataan vastaamaan koneoppimismallin ensimmäiselle konvoluutiokerrokselle asetettuja arvoja. Tämän jälkeen kuvan jokainen pikseliarvo käsitellään yksitellen. Yksittäinen pikseliarvo on RGB-muodossaan, joten jokainen värikanava täytyy vielä erottaa erilleen bittioperaatioilla. Lopuksi kanavat lisätään yksitellen ByteBuffer-olioon. (kuva 11.)

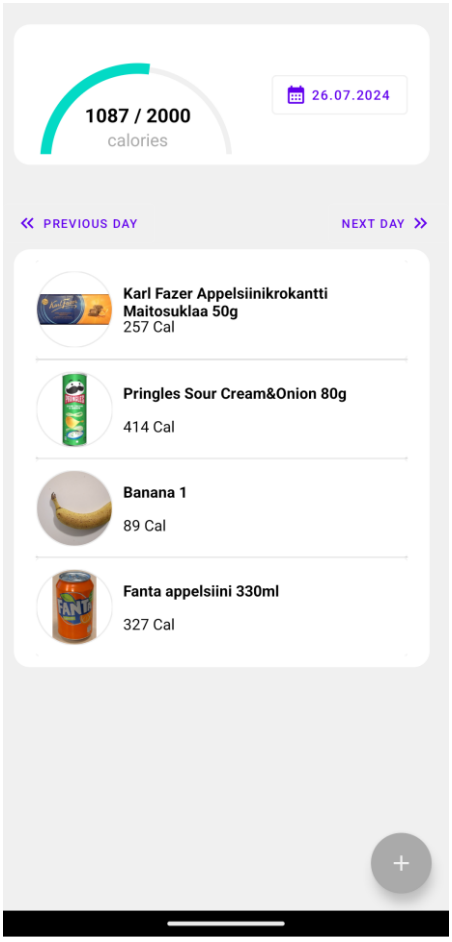
```
// Read all pixels into int array
final int[] pixelValues = new int[IMG_SIZE * IMG_SIZE];
img.getPixels(pixelValues,  offset: 0, img.getWidth(),  x: 0,  y: 0, img.getWidth(), img.getHeight());

// 4 bytes (float) * pixels * color channels
final ByteBuffer buffer = ByteBuffer.allocateDirect( capacity: 4 * IMG_SIZE * IMG_SIZE * 3);
buffer.order(ByteOrder.nativeOrder());

int currentPixel = 0;
for (int x = 0; x < IMG_SIZE; ++x) {
    for (int y = 0; y < IMG_SIZE; ++y) {
        final int rgbValue = pixelValues[currentPixel];
        buffer.putFloat( v: (rgbValue >> 16) & 0xFF);
        buffer.putFloat( v: (rgbValue >> 8) & 0xFF);
        buffer.putFloat( v: rgbValue & 0xFF);
        ++currentPixel;
    }
}
```

KUVA 11. Pikselien käsittely ennen kuvan tunnistusta

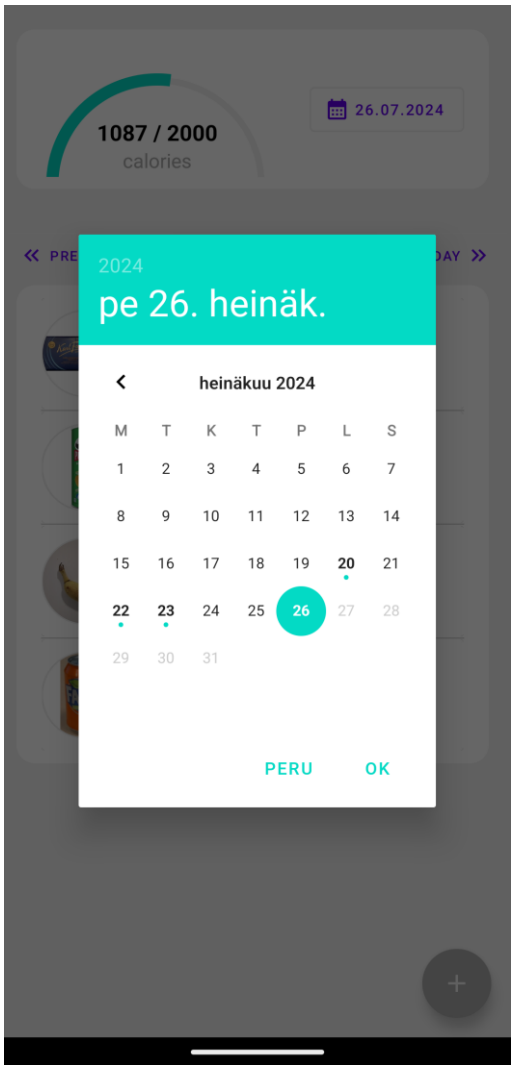
Tämän jälkeen ByteBuffer-olio ladataan TensorBuffer-oliolle, jota malli pystyy käsittelemään. Tunnistuksessa malli palauttaa toisen TensorBuffer-olion, joka sisältää lasketun todennäköisyyden jokaiselle kasvisluokalle taulukossa. Suurimman todennäköisyysarvon sisältävä luokka vastaa mallin mukaan kuvassa esitettyä kasvista.

4.2 Mobiilisovelluksen toteutus

Kuvassa 12 on sovelluksen aloitussivu. Sovellukseen lisätyt kasvikset ja tuotteet tallennetaan kuluvan päivän kokonaiskalorisaantiin. Lisätyistä tuotteista näkee kuvan ja nimen lisäksi määrän sekä lasketun kaloriarvon. Sovelluksella voidaan myös tarkastella muiden päivien kalorimääriä vaihtamalla päivämäärää kalenterin kautta (kuva 13) tai siirtymällä suoraan edelliseen tai seuraavaan päivään. Kalenteri korostaa niitä päiviä, jolloin käyttäjä on lisännyt tuotteita sen päivän kaloriseurantaan. Tulevia päiviä ei voi valita kalenterista.

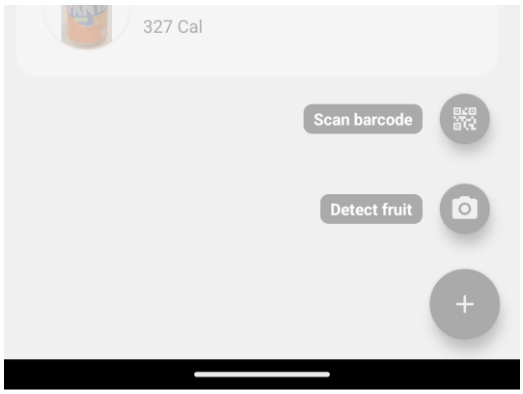


KUVA 12. Sovelluksen aloitussivu



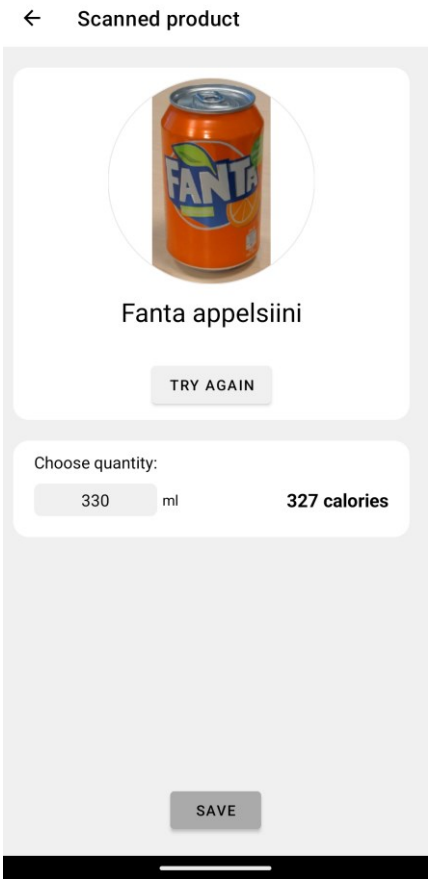
KUVA 13. Kalenterinäkymä

Sovelluksen valikon (kuva 14) kautta käyttäjä voi valita, haluaako hän lisätä kalorimäärän kasviksesta vai jostain muusta tuotteesta.



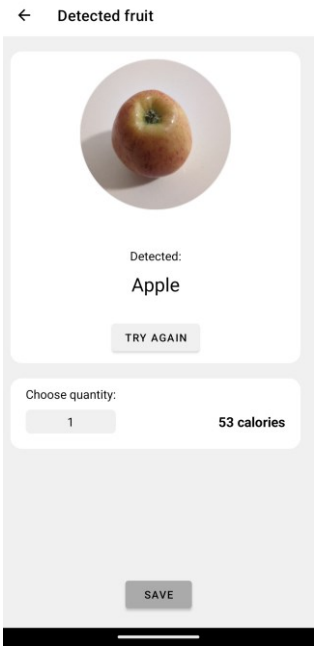
KUVA 14. Sovelluksen valikko

Molemmat vaihtoehdot avaavat mobiililaitteen kameran. Viivakoodin lukeminen käyttää Googlen GmsBarcodeScanner-koneoppimiskirjastoa, joka avaa laitteen kameran Google Play -palvelun kautta. Kirjasto on helppo ottaa käyttöön, eikä sen käyttö edellytä käyttöoikeuksien (engl. permissions) kyselyä käyttäjältä. Kirjasto tunnistaa viivakoodin tai QR-koodin välittömästi sen ilmestyessä kameran eteen ja palauttaa sitten koodissa olevan arvon. Viivakoodissa olevalla arvolla saadaan lopuksi haettua tuotetiedot Open Food Facts -sivustolta JSON-muodossa. Kuvassa 15 on luettu tuotteen viivakoodi.



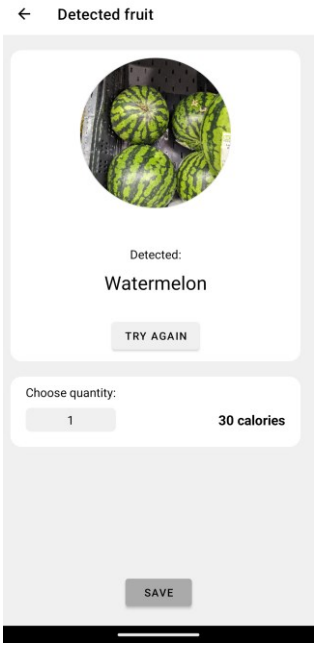
KUVA 15. Tuotteen lisääminen viivakoodilla

Kasviksen tunnistamisessa käyttäjä ottaa ensin kuvan kasviksesta tai valitsee kuvan laitteen galleriasta, jonka jälkeen koneoppimismalli analysoi kuvan ja pyrkii tunnistamaan kasviksen sekä hakemaan sen kalorimäärän CalorieNinjas-sivustolta. Kuvassa 16 ja 17 on esitetty eri kasvisten tunnistamista.



KUVA 16. Omenan tunnistaminen sovelluksessa

Ennen kasviksen tai tuotteen tallennusta käyttäjä pystyy valitsemaan määrän. Kasvisten osalta tämä ilmoitetaan kappalemääränä, nestemäisten tuotteiden kohdalla millilitroissa ja muiden tuotteiden osalta grammoissa. Määrän muuttaminen nollaksi tai muuten liian pieneksi arvoksi on estetty tekstisuodattimilla. Lopullinen kaloriarvo päivittyy reaaliajassa käyttäjän muuttaessa kappalemäärää ja laskentatapa riippuu tuotteen tyypistä. Tuotteen alkuperäinen kaloriarvo on ilmoitettu kasviksille per kappale, nestemäisille per 100 millilitraa ja muille tuotteille per 100 grammaa.



KUVA 17. Vesimelonin tunnistaminen sovelluksessa

Kaikki tuotteet tallennetaan paikalliseen SQLite-tietokantaan, joka on tiedosto käyttäjän mobiililaitteessa. Pieniin mobiilisovelluksiin ja webbisovelluksiin SQLite on ihan riittävä, mutta jos tämä sovellus tulisi julkiseen jakeluun, käyttäisin jotain ulkoista tietokantapalvelua kuten MongoDB:tä. SQLite on kuitenkin erittäin kevyt ja yksinkertainen, mikä tekee siitä helpon käyttää pienissä sovelluksissa (26).

SQLite-tietokantaa ei tarvitse myöskään asentaa erikseen tai se ei vaadi erillistä palvelinohjelmistoa, vaan se voidaan upottaa suoraan sovellukseen ja koko tietokanta on sisällytetty laitteella yhteen tiedostoon, jota on helppo siirtää ja varmuuskopioida (26). Tämä ominaisuus tekee SQLite:stä erityisen hyödyllisen mobiili- ja websovelluksille sekä IoT-laitteille, joissa optimaalinen suorituskyky ja alhaiset resurssivaatimukset ovat tärkeitä (27).

5 POHDINTA

Opinnäytetyön tavoitteena oli perehtyä tekoälytekniikkaan sekä sen erilaisiin alalajeihin, luoda tekoälymalli, joka kykenee tunnistamaan kasviksia kuvasta sekä tehdä mobiilisovellus, joka käyttää tätä mallia ja jolla voi seurata päivän kokonaiskalorisaantia. Koneoppiminen, neuroverkot ja tekoäly olivat minulle ennestään tuntemattomia aiheita, sillä en ollut koskaan aiemmin työskennellyt tekoälyn parissa enkä edes käyttänyt ChatGPT:n kaltaisia kielimalleja. Lähtökohtiin nähden opinnäytetyön tekeminen oli haastava, mutta samalla hyvin opettavainen ja mielenkiintoinen prosessi.

Koneoppimismallin sain lopulta toimimaan siten, että se pystyy tunnistamaan oikeita kasviksia kuvista. Suurimpana haasteena tämän tekemisessä oli kuitenkin huonolaatuiset kuvatietojoukot. Tällä hetkellä malli tunnistaa vain yhdeksää eri lajiketta, vaikka ensimmäisessä kuvatietojoukoissa oli alkujaan 36 erilaista lajia. Luomani testimalli antoi aluksi täysin selville hedelmän kuville virheellisiä vastauksia, jolloin luulin ongelman olevan neuroverkossani. Selvittelin ja säätelin kaikkia parametreja moneen kertaan, mutta mikään ei muuttunut. Etsin valmiita neuroverkko ratkaisuja internetistä, mutta malli ei edelleen kyennyt tunnistamaan esimerkiksi selvää banaania kuvasta. Kun kävin opetus-, validointi- ja testaustietojoukkoja läpi, huomasin että joukossa oli paljon huonoja kuvia. Poistamalla huonoimpia kuvia ja lisäämällä toisen kuvatietojoukon sain yhdelle lajikkeelle noin 500 opetuskuvaa. Tämän jälkeen malli alkoi tunnistamaan kuvia merkittävästi paremmin.

Mallin jatkokehityksenä voisi olla laadukkaamman ja suuremman kuvatietojoukon kasaaminen. Tietojoukko voisi sisältää myös kuvia, joissa esiintyy useita kappaleita samaa kasvista samassa kuvassa, kuten esimerkiksi banaaneja tertussa, koska neuroverkkomalli pystyy tällä hetkellä tunnistamaan vain yksittäisiä lajeja kuvasta. Lisäksi suurempi määrä opetuskuvia parantaisi mallin tarkkuutta ja luotettavuutta huomattavasti. Riittävä määrä opetuskuvia voisi olla tuhansia jokaisesta lajista.

Mobiilisovellusta yritin aluksi tehdä React Native -kielellä, mutta ainakaan opinnäytetyön kirjoitushetkellä sopivaa kirjastoa TensorFlow Lite -mallille ei löytynyt. GitHubin suosituin kirjasto ei jostain syystä pystynyt lukemaan tekemääni koneoppimismallia. Noin viikon selvittelyn jälkeen päätin vaihtaa tutumpaan Javaan ja Android Studioon. Mallin integrointi Android-sovellukseen oli todella suoraviivaista, eikä muissakaan sovelluksen osa-alueissa esiintynyt haasteita.

Sovelluksella pystyi siis lisäämään tunnistettuja hedelmiä ja vihanneksia kuluvan päivän kokonaiskalorisaantiin. Lisäksi sovelluksella voidaan lukea tuotteiden viivakoodeja, ja mikäli Open Food Facts -sivusto sisältää kyseisen tuotteen, sen kalorimäärä voidaan lisätä myös kalorisaantiin. Jatkokehityksenä suunnittelin, että koska kaikki tuotteet eivät tietenkään ole sivustolla saatavilla, loisin toisen koneoppimismallin, joka pystyisi tunnistamaan tuotteen kalorimäärän suoraan ravintosisällöstä.

Kokonaisuutena työ mielestäni onnistui, saavutin asetetut tavoitteet sekä opin todella paljon uutta tekoälystä ja sen alalajeista. Mobiilisovelluksen kehittäminen Javalla oli jo entuudestaan tuttua, mutta silti jokaisessa projektissa oppii aina jotain uutta. Esimerkiksi SQLite-tietokanta oli minulle tuttu muista ohjelmointikielistä, mutta en ollut aiemmin käyttänyt sitä Javalla.

LÄHTEET

1. UKK-instituutti 2022. Liikunta ja ravitsemus. Hakupäivä 30.07.2024. https://ukkinstituutti.fi/liike-laakkeena/liikunta-ja-ravitsemus/
2. European Parliament 2020. What is artificial intelligence and how is it used? Hakupäivä 21.05.2024. https://www.europarl.europa.eu/topics/en/article/20200827STO85804/what-is-artificial-intelligence-and-how-is-it-used
3. IBM Data and AI Team 2023. AI versus machine learning versus deep learning versus neural networks: What's the difference? IBM. Hakupäivä 23.05.2024. https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks
4. Glover, Ellen 2024. Artificial Intelligence. Built In. Hakupäivä 23.05.2024. https://builtin.com/artificial-intelligence
5. Cyber Chasse UK 2020. Facebook-päivitys 23.12.2020. Hakupäivä 24.05.2024. https://www.facebook.com/photo/?fbid=212540147113604
6. Thakur, Naresh 2020. The differences between Data Science, Artificial Intelligence, Machine Learning, and Deep Learning. Medium. Hakupäivä 27.05.2024. https://ai.plainenglish.io/data-science-vs-artificial-intelligence-vs-machine-learning-vs-deep-learning-50d3718d51e5
7. SAP 2024. What is machine learning? Hakupäivä 27.05.2024. https://www.sap.com/products/artificial-intelligence/what-is-machine-learning.html
8. MinnaLearn 2024. Mitä on tekoäly ja koneoppiminen. Hakupäivä 27.05.2024. https://courses.minnalearn.com/fi/courses/ai-for-built-environment/tekoalyn-hyodyntamisen-edellytykset/mita-on-tekoaly-ja-koneoppiminen/
9. Holdsworth, Jim & Scapicchio, Mark 2024. What is deep learning? IBM. Hakupäivä 14.07.2024. https://www.ibm.com/topics/deep-learning
10. IBM 2024. What is a neural network? Hakupäivä 16.07.2024. https://www.ibm.com/topics/neural-networks
11. Cloudflare 2024. What is a neural network? Hakupäivä 16.07.2024. https://www.cloudflare.com/learning/ai/what-is-neural-network/
12. Amzhao 2023. Quantum Neural Networks. Medium. Hakupäivä 16.07.2024. https://medium.com/mit-6-s089-intro-to-quantum-computing/quantum-neural-networks-7b5bc469d984

13. Baheti, Pragati 2021. Activation Functions in Neural Networks. V7. Hakupäivä 16.07.2024. https://www.v7labs.com/blog/neural-networks-activation-functions
14. IBM 2024. What are convolutional neural networks? Hakupäivä 17.07.2024. https://www.ibm.com/topics/convolutional-neural-networks
15. Kh Nafizul Haque 2023. LinkedIn-päivitys 03.04.2023. Hakupäivä 17.07.2024. https://www.linkedin.com/pulse/what-convolutional-neural-network-cnn-deep-learning-nafiz-shahriar
16. Brownlee, Jason 2019. A Gentle Introduction to Pooling Layers for Convolutional Neural Networks. Machine Learning Mastery. Hakupäivä 17.07.2024. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/
17. Mandal, Manav 2024. Introduction to Convolutional Neural Networks (CNN). Analytics Vidhya. Hakupäivä 17.07.2024. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
18. Databricks 2024. TensorFlow. Hakupäivä 14.05.2024. https://www.databricks.com/glossary/tensorflow-guide
19. Google Colaboratory 2024. Mikä Colab on? Hakupäivä 15.05.2024. https://colab.research.google.com/
20. Nvidia 2024. TensorFlow. Hakupäivä 16.05.2024. https://www.nvidia.com/en-us/glossary/tensorflow/
21. Databricks 2024. Keras Model. Hakupäivä 28.07.2024. https://www.databricks.com/glossary/keras-model
22. Seth, Kritik 2020. Fruits and Vegetables Image Recognition Dataset. Kaggle. Hakupäivä 05.05.2024. https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition/
23. Oltean, Mihai 2017. Fruits-360 dataset. Kaggle. Hakupäivä 09.05.2024. https://www.kaggle.com/datasets/moltean/fruits/data
24. Chollet, François 2023. Training & evaluation with the built-in methods. TensorFlow. Hakupäivä 17.05.2024. https://www.tensorflow.org/guide/keras/training_with_built_in_methods
25. Tyagi, Varun 2024. Unraveling the Mysteries of Image Recognition with Convolutional Neural Networks (CNN) in Python. Medium. Hakupäivä 27.07.2024. https://medium.com/@varun.tyagi83/unraveling-the-mysteries-of-image-recognition-with-convolutional-neural-networks-cnn-in-python-83b3e5b75ef3
26. SQLite 2023. About SQLite. Hakupäivä 27.07.2024. https://www.sqlite.org/about.html

27. SQLite 2024. Appropriate Uses For SQLite. Hakupäivä 27.07.2024. https://www.sqlite.org/whentouse.html