Data-Ai Kurssi
Oppimistehtävä 2

Victor Stén

Pistearvio 1

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

Datan valitseminen

Tässä oppimistehtävässä pohdin aluksi pitkään, minkä aiheen haluaisin valita. Viimeisimmässä CNN-tehtävässä innostuin kuitenkin kuvatunnistuksesta, joten päätin perustaa myös tämän toisen oppimistehtävän samalle aihealueelle.

Tehtävä alkoi erilaisten datasetien tarkastelulla: mitä haluaisin opettaa konetta tunnistamaan koneoppimisen keinoin. Selailin Kaggle-nimiseltä sivustolta useita datalähteitä aiheina muun muassa kukkien ja eläinten tunnistus, kunnes lopulta päädyin hedelmiin liittyvään aineistoon.

Lopullinen valintani oli Fruits-360-niminen datasetti, joka sisältää noin 120 030 kuvaa 176 eri luokasta. Mukana on hedelmiä, vihanneksia, pähkinöitä ja siemeniä. Datasetti oli yllättävän kevyt kooltaan, vain noin 2,75 Gt, mikä johtuu siitä, että kuvat ovat kooltaan 100x100 pikseliä. Tämä pienempi kuvatarkkuus vähentää mallin käsittelemää datamäärää ja nopeuttaa merkittävästi mallin koulutusaikaa – mikä on erittäin toivottavaa.

Kun datasetti oli valittu, tutustuin ensin sen dokumentaatioon eli tarkastelin mitä kaikkea aineisto sisältää ja mitä sen laatija on dokumentoinut käytöstä. Datasetissä on kaksi versiota:

1. 100x100-resoluutiolla tiivistetyt kuvat, joita on eniten,
2. "Original-size branch", jossa on noin 19 852 kuvaa alkuperäisillä ja vaihtelevilla resoluutioilla.

Dokumentaatiossa kerrotaan myös, miten kuvat on tuotettu. Hedelmät ja vihannekset on asetettu pyörivälle alustalle, joka liikkui noin 3 kierrosta minuutissa. Tästä otettiin 20 sekunnin mittainen video Logitech C920 -webkameralla. Taustana käytettiin valkoista paperia, ja lopuksi kuvat sekä taustan erotus hoidettiin automaattisesti.

Kurssilla huomautettiin asiasta, että jos tekee koneoppimista kuvien kanssa niin ideaalista on käyttää näytönohjaimen tehoja prosessorin sijasta. Se nopeuttaa merkittävästi itse koneoppimista. Seuraavana vaiheena on itse koulutus koodin rakentaminen. Ajattelen, että ensimmäisenä voisi aloittaa kahden hedelmän tunnistamisesta joten nähdään miten malli reagoi.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

Koodin kirjoittaminen ja selittäminen

Ensimmäisenä tavoitteena asetetaan kahden hedelmän tunnistus. Tässä tapauksessa banaani ja kurkku. Koodin rakentaminen tulee ensimmäisenä aloittaa, että koulutus ja testidata ovat ainoastaan banaaniin ja kurkkuun liittyviä. Sitä varten kopioin banaanin ja kurkun testi ja treenaus datan sen omaan kansioon. Fruits-360 on erikoisena se, että siellä on valmiiksi testi ja treenausdatat eriteltynä omiin kansioihin, joten meidän ei tarvitse erikseen tehdä sitä.

Alkuperäisessä testi tehtävässä jonka teimme tunnilla data oli jo valmiiksi käsitelty Mnistin kautta. Nyt meidän pitää itse käsitellä se kerasin.preprocessing imageDataGeneraattorin kautta. Se on kerasin työkalu, joka lukee kuvia kansioista jonka mukaan kuvat skaalataan niitten resoluution avulla. Se myös luokittelee kuvat kansioiden nimistä esimerkiksi nyt banaanin ja kurkun mukaisesti.

```
# Polut
train_dir = 'D:/Koulujutut/25Kevat/PROJEKTIT/Koneoppiminen/banana-cucumber/Training'
test_dir = 'D:/Koulujutut/25Kevat/PROJEKTIT/Koneoppiminen/banana-cucumber/Test'

# Kuvien koko
img_width, img_height = 100, 100
input_shape = (img_width, img_height, 3)

# Datan esikäsittely
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=32,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_width, img_height),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

num_classes = train_generator.num_classes
```

Kuva 1 Mallin koulutus koodipätkä 1

Datan esikäsittelyssä on tärkeätä, että kuvien pikseliarvot skaalataan neuroverkkoja varten sopivaksi. Tässä tapauksessa ne skaalataan desimaaliluvuiksi välille 0-1 jotta neuroverkko oppii nopeammin mitä pienemmät arvot ovat.

Kuvat ladataan kansiorakenteesta flow_from_directory()-funktiolla. Jokainen alikansio (esim. *Banana*, *Cucumber*) vastaa yhtä luokkaa, ja ne muunnetaan automaattisesti oikeaan muotoon ja kokoon. Testidatan kohdalla shuffle=False pitää kuvien järjestyksen samana, jotta arviointi (esim. confusion matrix) toimii oikein.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

```
# CNN-malli
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',padding='same', input_shape=input_shape))
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(num_classes, activation='softmax'))
```

Kuva 2 Mallin koulutus koodipätkä 2

Seuraavaksi käytetään itse CNN-mallia. Tämä rakenne on samanlainen kuin alkuperäisessä numeron tunnistukseen liittyneessä CNN-tehtävässä. Mallin toimintaa testataan aluksi yksinkertaisemmalla versiolla, jotta nähdään, kuinka hyvin se suoriutuu ennen kuin siihen lisätään uusia kerroksia.

Ensimmäiset kaksi kerrosta ovat konvoluutiokerroksia, jotka etsivät kuvista muotoja, reunoja ja kulmia. Tämän jälkeen *MaxPooling2D* pienentää kuvan kokoa, mikä nopeuttaa mallin toimintaa ja vähentää muistinkulutusta. *Flatten()*-funktio muuntaa kahden ulottuvuuden kuvan yhdeksi pitkäksi vektoriksi, joka voidaan syöttää tiheille (Dense) kerroksille.

Dense(128)-kerros käsittelee opitut piirteet ja yhdistää ne, ja viimeisenä *Dense(num_classes, activation='softmax')* palauttaa todennäköisyydet jokaiselle luokalle. Lopullinen ennuste on se luokka, jolla on suurin todennäköisyys.

Tässä vaiheessa käytetään vielä kevyttä mallia, ja mikäli tulokset eivät ole riittävän hyviä, voidaan lisätä lisää konvoluutiokerroksia mallin syvyyden kasvattamiseksi.

```
# Mallin Käännös
opt = SGD(learning_rate=0.001, momentum=0.9)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])

# Visualisointi
plot_model(model, to_file='banana_cucumber_model.png', show_shapes=True, show_layer_names=True)

# Koulutus
history = model.fit(train_generator, epochs=20, validation_data=test_generator)

# Arviointi
loss, acc = model.evaluate(test_generator, verbose=1)
print(f"\nTest Loss: {loss}")
print(f"Test Accuracy: {acc}\n")
```

Kuva 3 Mallin koulutus koodipätkä 3

Seuraavaksi määritellään optimointialgoritmi, joka tässä tapauksessa on *SGD (Stochastic Gradient Descent)*. Samaa algoritmia käytettiin myös alkuperäisessä numerotunnistustehtävässä. SGD on yksi yleisimmistä optimointialgoritmeista, joita käytetään neuroverkoissa, ja se toimii hyvin erityisesti pienemmissä tai selkeissä kuvatehtävissä.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

Tässä vaiheessa määritellään myös mallin tarkkuuden seuranta käyttämällä **"accuracy"**-mittaria. Tämä mittari näkyy koulutuksen aikana terminaalissa ja kertoo, kuinka monta ennustetta malli teki oikein.

Lopuksi suoritetaan mallin **kääntäminen** (compile), jossa määritellään, miten malli oppii. Tässä vaiheessa asetetaan kolme tärkeää asiaa:
– häviöfunktio (loss)
– optimointialgoritmi (optimizer)
– seurattava metriikka (metrics)

Käytetty häviöfunktio on **categorical_crossentropy**, jota käytetään, kun luokkia on useita ja ne on one-hot-koodattuja. Se mittaa, kuinka kaukana mallin ennusteet ovat oikeasta vastauksesta, ja auttaa ohjaamaan mallin oppimista kohti tarkempia tuloksia.

```
# Koulutus
history = model.fit(train_generator, epochs=20, validation_data=test_generator)

# Arviointi
loss, acc = model.evaluate(test_generator, verbose=1)
print(f"\nTest Loss: {loss}")
print(f"Test Accuracy: {acc}\n")

# Oppimiskäyrät
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()

# Ennusteet ja tarkkuus
y_pred_proba = model.predict(test_generator)
y_pred = np.argmax(y_pred_proba, axis=1)
y_true = test_generator.classes

cm = confusion_matrix(y_true, y_pred)
acc = accuracy_score(y_true, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='g', cmap="Blues", xticklabels=class_names, yticklabels=class_names)
plt.title(f'Confusion Matrix (Accuracy: {acc:.2f})')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.tight_layout()
plt.show()

print("Classification Report:\n")
print(classification_report(y_true, y_pred, target_names=class_names))

# Mallin tallennus
model.save("banana_cucumber_model.keras")
```

Kuva 4 Mallin koulutus koodipätkä 4

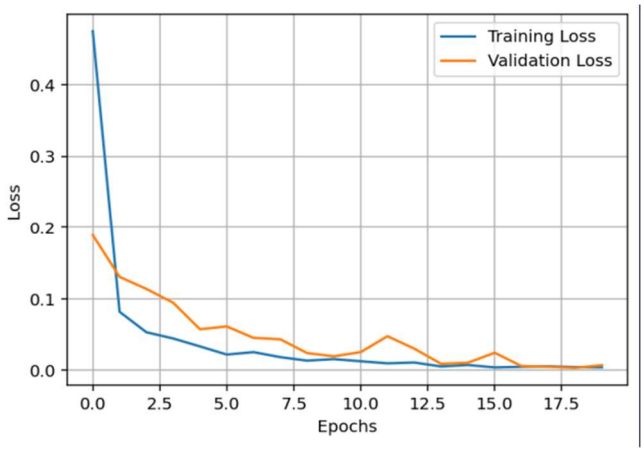
Seuraavat vaiheet ovat enemmän kuvaajien, oppimiskäyrien, metriikoiden ja ennusteiden tarkkuuksien visualisointia ja itse mallin tallentamista varten.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

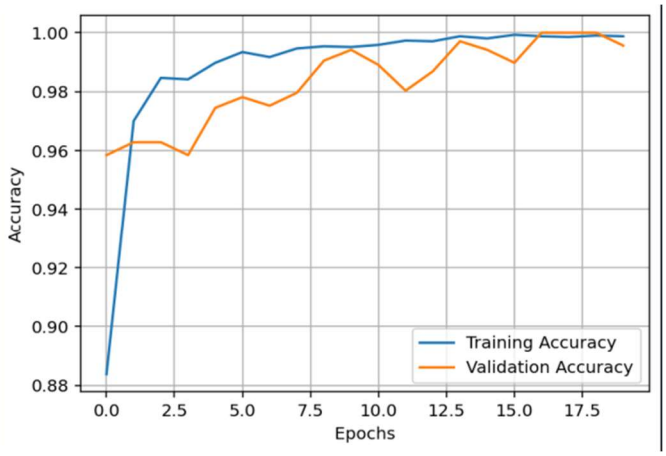
Mallinnuksen kouluttaminen ja testaaminen

Seuraavaksi aloitetaan itse mallinusten kouluttaminen ja niiden testaaminen. Ensimmäisenä aloitetaan pienemmällä koulutusmateriaalilla eli kurkulla ja banaanilla. Mallinnuksen tavoite on siis yksinkertaisesti erottaa annetusta kuvasta onko se banaani vai kurkku. Mallinnuksen koulutus aloitetaan ensimmäisenä yksinkertaisemmalla CNN-mallilla jonka pitäisi pystyä tunnistamaan kuvat oikein.

1. Testi (Banaani ja kurkku)
   Mallin koulutus tehtiin noin 20 epokilla, jonka koulutus kesti noin 3.5 min.



Kuva 6 Oppimiskäyrä (loss)



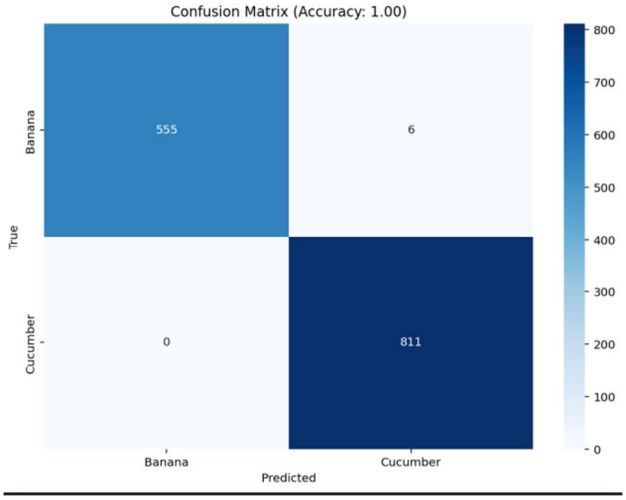
Kuva 5 Oppimiskäyrä (Accuracy)

Yllä olevissa kuvaajissa esitetään mallin oppimisen eteneminen. Vasemmalla puolella näkyy loss-käyrä, joka kertoo, kuinka hyvin malli oppii pienentämään virhettä harjoitus- ja validointidatassa. Oikealla puolella näkyy accuracy-käyrä, joka kertoo mallin tarkkuuden kehityksestä.

Molemmat kuvaajat osoittavat, että malli oppii nopeasti jo ensimmäisten epookkien aikana. Harjoitus- ja validointihäviöt pienenevät tasaisesti, ja tarkkuudet nousevat lähelle 100 %. Malli ei myöskään näytä ylioppivan, koska validointikäyrät pysyvät lähellä harjoituskäyriä koko koulutuksen ajan.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

Mallinnuksesta saadaan myös laadittua confusion matrix, jonka avulla nähdään, kuinka hyvin malli osasi luokitella eri luokat oikein. Tässä tapauksessa malli oppi tunnistamaan banaanit ja kurkut erittäin hyvin – ainoastaan 6 banaania luokiteltiin virheellisesti kurkuksi.



Kuva 7 Mallinnus 1 Confusion matrix

Seuraavaksi testataan itse mallin toimivuutta tekemällä erillinen testaus tiedosto, joka tehdyn mallin avulla arvioi kolmesta eri kuvasta onko kyseessä kurkku vai banaani. Tämä perustuu samaan koodinpätkään mitä käytettiin myös numeroiden ennustamisessa.

```python
from keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
import os

# Ladataan malli
model = load_model('banana_cucumber_model.keras')

# Haetaan luokkien nimet samassa järjestyksessä kuin koulutuksessa
temp_datagen = ImageDataGenerator(rescale=1./255)
temp_generator = temp_datagen.flow_from_directory(
    'D:/Koulujutut/25Kevat/PROJEKTIT/Koneoppiminen/banana-cucumber/Training',
    target_size=(100, 100),
    batch_size=1,
    class_mode='categorical'
)
class_names = list(temp_generator.class_indices.keys())

# Testataan yksittäisiä kuvia samasta kansiosta.

for i in range(0, 3):  # testataan 0.jpg, 1.jpg, 2.jpg
    file = f'{i}.jpg'

    if not os.path.exists(file):
        print(f"Tiedostoa ei löydy: {file}")
        continue

    # Ladataan ja esikäsitellään kuva
    image = load_img(file, target_size=(100, 100))  # RGB oletuksena
    image_array = img_to_array(image)
    image_array = image_array.reshape(1, 100, 100, 3).astype('float32') / 255.0

    # Ennustus
    pred_proba = model.predict(image_array)
    pred_class = np.argmax(pred_proba)
    confidence = pred_proba[0][pred_class]

    # Näytetään kuva ja tulokset
    plt.imshow(image)
    plt.axis('off')
    plt.title(f'{file} ➜ {class_names[pred_class]} ({confidence:.2f})')
    plt.show()

    print(f'File: {file} ➜ Predicted class: {class_names[pred_class]} (confidence: {confidence:.2f})')
```

Kuva 8 Testaus tiedosto

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

Seuraavaksi testaamme kuinka malli tunnistaa kuvista onko kyseessä kurkku vai banaani. Testaukseen käytettiin 4 eri kuvaa joista kaksi olivat banaaneja ensimmäinen erittäin kypsä ja toinen raaka. Raakaa banaania yritin saada enemmän testaamaan meneekö malli harhaan raan banaanin väristä. Kaksi muuta kuvaa olivat avomaankurkku eli lyhyempi ja normaali kaupasta saatava pidempi kurkku.

```
Found 4105 images belonging to 2 classes.
1/1 [==============================] - 0s 33ms/step
File: 0.jpg ➜ Predicted class: Banana (confidence: 1.00)
1/1 [==============================] - 0s 17ms/step
File: 1.jpg ➜ Predicted class: Cucumber (confidence: 0.64)
1/1 [==============================] - 0s 17ms/step
File: 2.jpg ➜ Predicted class: Banana (confidence: 1.00)
1/1 [==============================] - 0s 17ms/step
File: 3.jpg ➜ Predicted class: Banana (confidence: 1.00)
```

Tuloksissa malli ennusti neljästä testikuvasta kolme oikein ja yhden väärin. Virheellinen luokittelu saattoi johtua siitä, että käytetyssä datasetissä kurkun kuvat ovat pääasiassa avomaankurkkuja, jotka ovat lyhyempiä ja pyöreämpiä kuin kaupasta saatavat tavalliset kurkut. Tämän vuoksi malli on voinut sekoittaa kurkun banaaniin muodon perusteella.

On myös hyvä huomioida, että vaikka yksi kurkku ennustettiin oikein, mallin varmuus oli vain 64 %, mikä kertoo epävarmuudesta luokittelussa. Vaikka tulos ei ole täydellinen, se osoittaa, että malli on jo oppinut erottamaan luokkia, mutta vielä on tilaa parannukselle.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

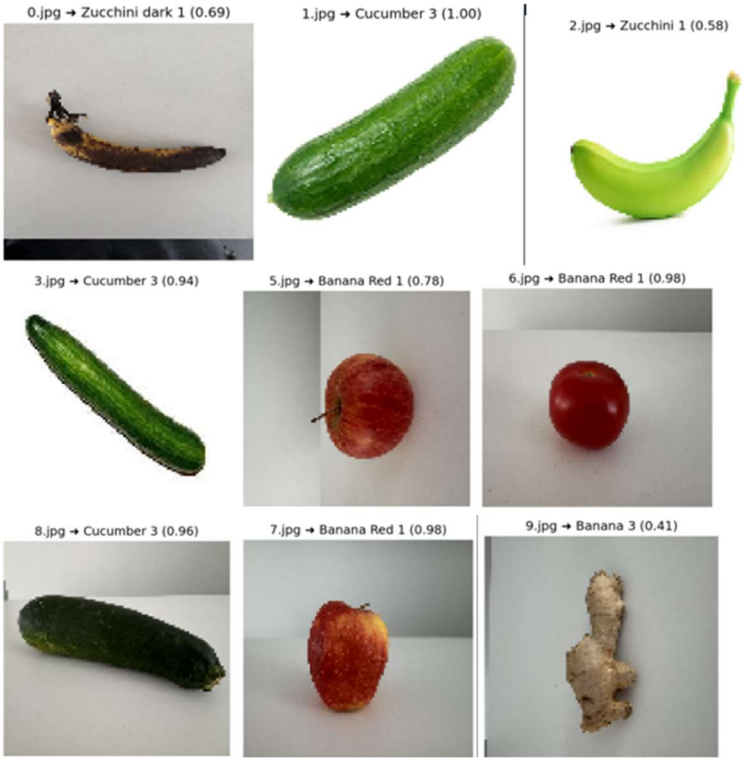
Testi 2: koko datasetti

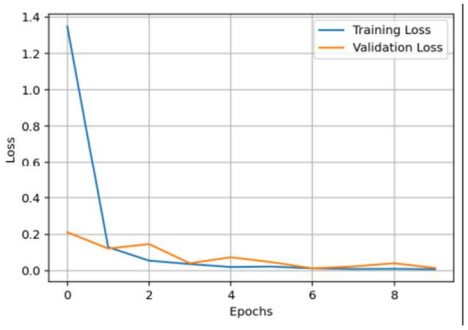
Aloitin testaamisen suoraan itse koko käytettävästä datasetistä josta olin korjannut pois sellaiset hedelmät / vihannekset mitä nyt omasta vihannes/hedelmälaatikosta kohtaa. Pääsääntöisesti siis yleisimmät mitä suomessa esiintyy kaupassa.

```
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(img_width, img_height),
    batch_size=32,
    color_mode='rgb',
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=(img_width, img_height),
    batch_size=32,
    color_mode='rgb',
    class_mode='categorical',
    shuffle=False
)
```

Koodi oli kokonaisuudessaan sama poislukien muutin datan esikäsittelyyn lisäksi color_mode rgb, joka siis määrittää missä väriformaatissa kuva ladataan ja annetaan mallille. Tämä olisi oleellinen sen suhteen, että malli ei ainoastaan ennustaisi muotojen perusteella.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén
Lopuksi sain tämän lopputuloksen





Kokonaisuudessa malli ennusti aika huonosti se meni hieman sekasin kaikkien osalta mutta kurkut se osasi arvioida oikein. Tämän pohjalta lähdin kokeilemaan muuttaen hieman mallia ja kokeillen sellaisia optimoijia kuin RMSprop ja Adam. Myös malliin lisäsin enemmän kerroksia.

```
# CNN-malli
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D())
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

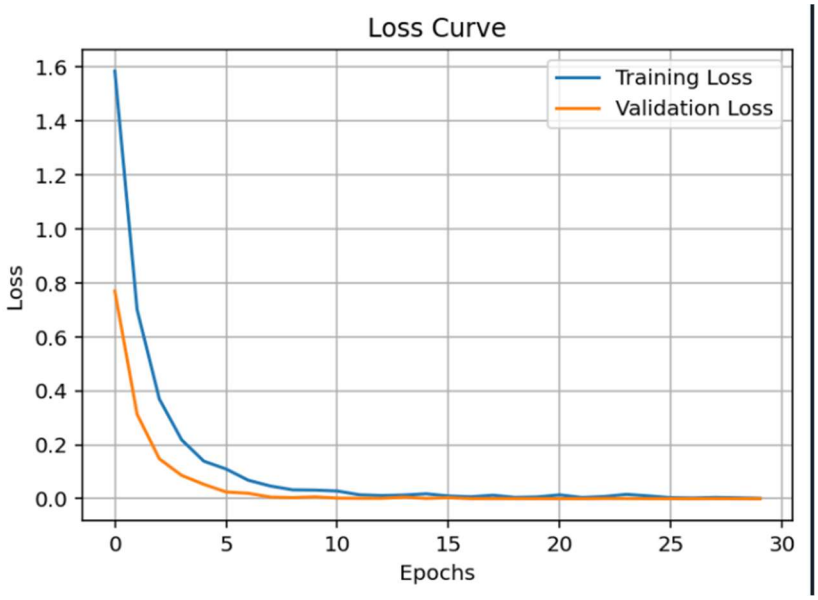
Testasin useita eri rakenteisia malleja, mutta tulokset pysyivät pääosin samalla tasolla, eikä merkittäviä parannuksia havaittu. Tarkastelin myös muiden toteutuksia Kaggle-alustalla ja vertailin lähestymistapoja omaani. Huomasin, että useimmat käyttivät RMSprop-optimointia ja validointidatana samoja kuvia kuin alkuperäisessä datasetissä.

Itse käytin pääasiassa omia ottamiani kuvia sekä myöhemmin myös internetistä haettuja kuvia hedelmistä ja vihanneksista valkoisella taustalla. Kävi selväksi, että kuvat tunnistettiin huomattavasti paremmin silloin, kun tausta oli täysin valkoinen ja varjoton.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

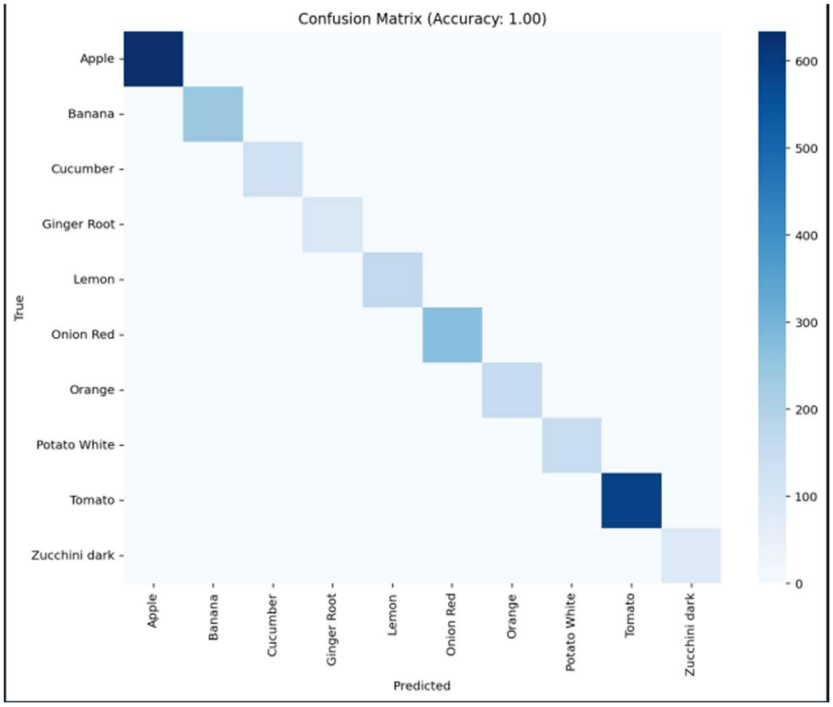
Testailin eri mallivariaatioita koko päivän ajan, mutta en saanut aikaiseksi merkittäviä tulosparannuksia. Lopullisessa testissä käytin mallia, jossa optimointiin oli valittu Adam-algoritmi. Tätä kokeilin vielä viimeisenä vaihtoehtona.

```
# CNN-malli
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D())
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Mallin käännös
opt = Adam(learning_rate=0.0001)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

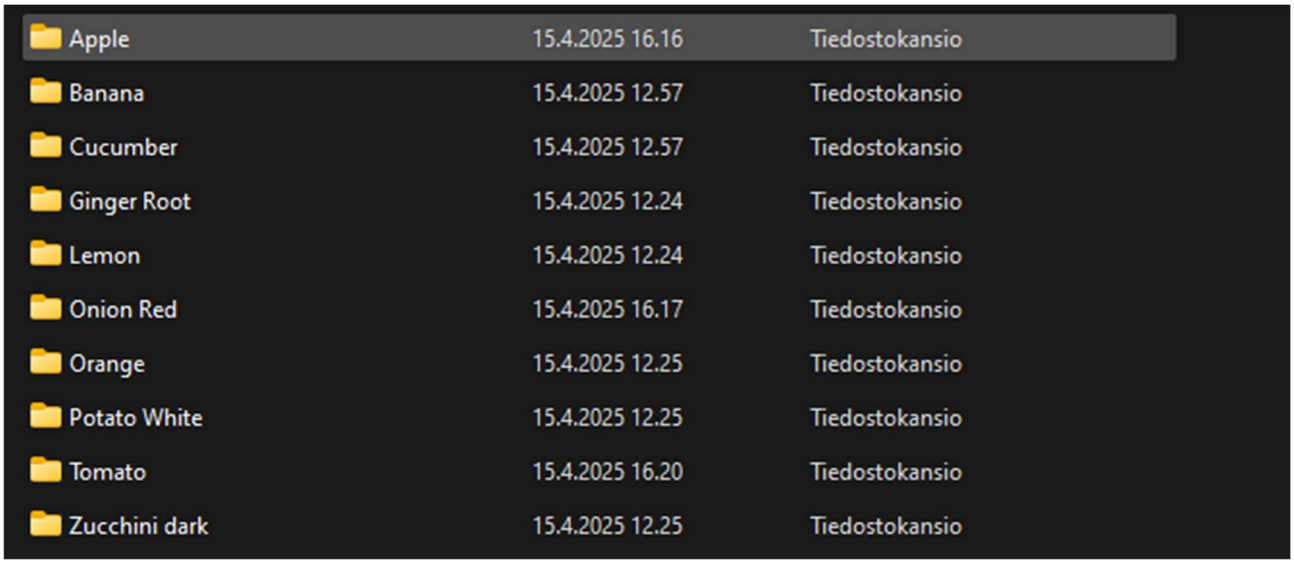


Malli oppi melko nopeasti, eikä ylioppimista näyttänyt syntyvän ainakaan oppimiskäyrien perusteella. Testasin myös vastaavanlaisia malleja, joissa käytin Dropout-kerroksia ja joiden

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén
koulutus kesti noin kaksi tuntia, mutta ne tuottivat hyvin samankaltaisia tuloksia.

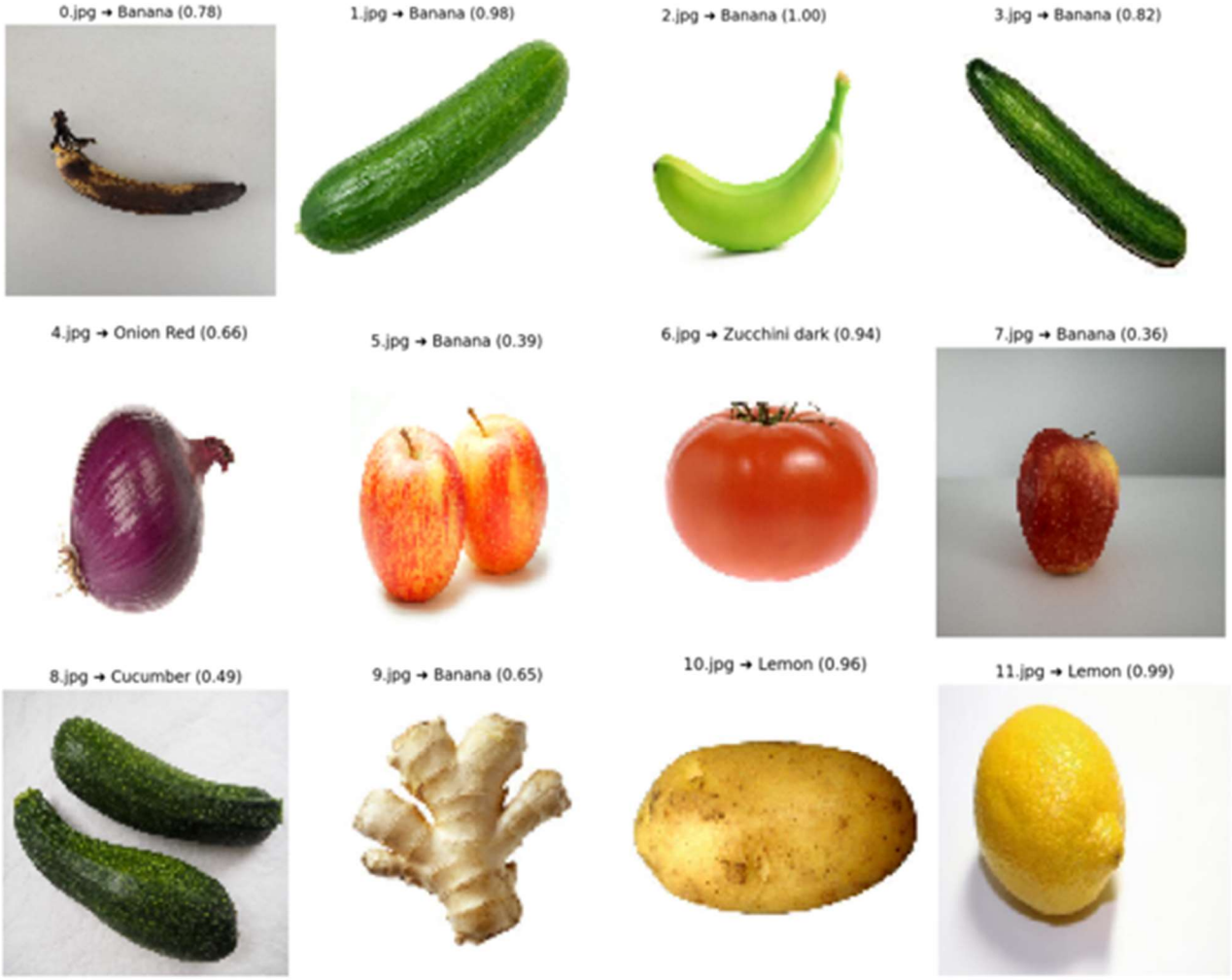


Tästä confusion matrixista voidaan erityisesti huomata, kuinka epätasaisesti dataa on eri hedelmistä ja vihanneksista. Esimerkiksi omenoista ja tomaateista oli mukana useita erilaisia versioita, mikä näkyy vahvasti myös kuvaajassa. Yritin vähentää näiden ylikorostumista, mutta malli nojasi silti voimakkaasti juuri näihin luokkiin. Mielestäni datasetissä tulisi olla mahdollisimman tasainen määrä kuvia kaikista luokista, jotta tällaisia vinoumia ei syntyisi mallin oppimisessa.



Tässä on listattuna kaikki Koulutusdataan ja testidatassa olevat vihannekset/Hedelmät

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén



Lopputuloksena voidaan todeta, että malli tunnisti joitakin yksittäisiä kuvia oikein, mutta suurin osa jäi virheellisesti luokitelluiksi. Parhaimmillaan pääsin tuloksiin, joissa noin neljä kuvaa oli luokiteltu oikein. Testauksen aikana kävi selvästi ilmi, että malli 'tarrautui' helposti tiettyyn hedelmään tai vihannekseen yleensä siihen, josta datasetissä oli eniten esimerkkejä. Tämä johti siihen, että suuri osa testikuvista luokiteltiin väärin samaksi luokaksi.

Positiivista kuitenkin oli, että malli onnistui tunnistamaan edes joitakin kuvia oikein. Erityisesti punasipuli yllätti: aiemmat mallit luokittelivat sen toistuvasti kesäkurpitsaksi, mutta nyt se tunnistettiin oikein useammin.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén

Oman työn arviointi: onnistumiset ja kehityskohteet sekä arvio pistemäärästä (1 tai 2)

Kokonaisuudessaan projekti oli todella mielenkiintoinen ja innostava. Aihe jaksoi motivoida jatkuvaan testaamiseen ja uusien lähestymistapojen kokeiluun. Vaikka hedelmien ja vihannesten tunnistaminen kuulostaa yksinkertaiselta tehtävältä, osoittautui se varsin haastavaksi erityisesti silloin, kun kuvat olivat itse otettuja ja käytettiin rajallista määrää dataa.

Kuvien määrä osoittautui yhdeksi suurimmista rajoitteista. Vaikka valkoinen tausta paransi tunnistustarkkuutta selvästi, olisi tarvittu vielä huomattavasti enemmän koulutusmateriaalia, jotta malli oppisi erottamaan eri luokat luotettavasti. Tämä korostui etenkin itse otetuissa tai netistä kerätyissä kuvissa, joissa valaistus, tarkennus ja taustat vaihtelivat enemmän kuin alkuperäisessä datasetissä.

Myös itse datasetin rakenne aiheutti haasteita. Esimerkiksi omenista oli mukana noin 19 eri lajiketta, joilla jokaisella oli oma kansionsa. Tämä johti siihen, että omenoista kertyi valtava määrä esimerkkejä, mikä puolestaan aiheutti sen, että malli "biasoitui" ja luokitteli suuren osan testikuvista omenoiksi tai muuksi. Tämän vuoksi jouduin yksinkertaistamaan datasettiä yhdistämällä useita omenaluokkia ja muita hedelmiä/vihanneksia yhdeksi yhteiseksi luokaksi, mutta se ei täysin ratkaissut ongelmaa.

Testausta tuli tehtyä paljon eri mallirakenteilla ja optimointimenetelmillä, mutta lopputulos jäi osittain vajaaksi. Parasta mahdollista mallia en ehtinyt kehittää, osittain muiden yhtäaikaisten projektien vuoksi. Siitä huolimatta tehtävä oli erittäin antoisa ja kehittävä sekä teknisesti että oppimiskokemuksena.

Lopulliseksi pistemääräksi arvioin 1.

Data-Ai Kurssi
Oppimistehtävä 2
Victor Stén
Datalähde: https://www.kaggle.com/datasets/moltean/fruits/data

Liitte1: Oppimistehtävä2.py

Liite 2: Fruits_test.py