Project Report

Course: Multimedia Retrieval

Student: Amir Dohmosh

Matriculation number: 107209

Topic: Classification of fruits and their freshness

1. The Aim of the Project

The aim of this project is to develop a system that can identify specific fruit and its freshness. I would be training a convolutional neural network (CNN) to be used to solve this problem. For this purpose, around 10,000 fruit images are used for the training of the network and around 3,000 images are used for the testing of the network.

This project would hopefully create a way to help in saving time and effort in the sorting process at supermarkets and also help to a certain level in the inspection and evaluation the freshness of the fruit.

2. Software Tools

A number of tools and libraries were needed for the different stages of the implementation of my system. There were major tools such as the Flask for implementing the server side, HTML5, CSS3 and JavaScript on the front-end side, TensorFlow for building and training the model, and other small libraries like matplotlib for displaying images. I would be briefly discussing some tools and the reasons they were chosen.

First of all, I decided to write the system in python programming language, because it is one of the simplest programming languages and is easy to use with TensorFlow. Also, there are a number of python libraries that would be helpful for my objective.

- Matplotlib

  Matplotlib is a python library that can be used to create visual representation of mathematical data or simply static or animated visuals. It also has an object-oriented API that allows you to embed these image representations into other applications. It can be used to make line plots, histograms, scatter plots, polar plots, 3D plots, etc. It is an open source software and therefore has very little restriction on its use. Within my system I would be using matplotlib to display images of fruits where necessary and to plot the image histogram representation of the fruit images.

- Numpy

  Numpy is a python library used for making, processing and operating on matrices and multidimensional arrays. This would be used to handle and process all data of the array data type within the project and also to help with the convolution and and pooling operations on the data

- TensorFlow

  TensorFlow is an open source software library for machine learning used in Google products. TensorFlow library combines various APIs to create a deep learning architecture like CNN (Convolutional Neural Network) or RNN (Recurrent Neural Networks). TensorFlow Architecture is based on graph computation. Currently, TensorFlow is used in many fields such as voice recognition, computer vision, robotics, information retrieval, and natural language processing.

- Flask

  Flask is a microframework with little to nothing in its box besides the framework itself. The benefit of this is the framework's lightweight build. It does not require tools or libraries, encouraging the development of simple web applications.

3. CNN Implementation

The network itself is implemented using transfer learning. The MobileNetV2 is a convoluted Neural Network that can also be used for our freshness recognition application. The unique property of this network is its ability to achieve good image recognition performance while at the same time reduce the number of calculations needed to execute the model. The MobileNetV2 model developed at Google is used as a base model for feature extraction from our data. A custom classification layer is added on top and trained separately.

The process of using the MobileNetV2 may be more beneficial for our application as the reduction in the number of computations will help the application run better in mobile applications where the amount of computational power is limited.

There are several ways to train a model using transfer learning. For my project, I first loaded the pre-trained model and discarded the last layer. This is a dense layer with 1000 neurons that serves as a classifier of the previous feature map. Once discarded, I set the rest of the layers to not trainable (this prevents the weights in a given layer from being updated). Then, at the end of the network I added another dense layer, but now with the number of classes of

fruits I want to predict. After that, I trained the model for 20 epochs. I use batch normalization after every layer. The base learning rate is set to 1e −4 and a batch size set to 32.

4. Implementation and Training of the CNN Model

4.1. Dataset

Data is an essential part of Deep Learning. Therefore, it is important to select the correct input data according to our goals. For fruit classification, there exists a dataset called Fruits fresh and rotten for classification, which consists of 13,599 images belonging to 6 classes of fruits. A major drawback of this dataset is that the images have no background, thus it does not scale very well to real-world applications.

For my project, I use 6 classes of fruits (fresh/rotten apples, fresh/rotten oranges and fresh/rotten bananas). I split the dataset into two groups for training and testing, with 10,901 images for training and 2,698 images for testing.

Table 4.1. Number of images

Classes	**Images**	**Testing Images**	**Training Images**
Fresh apples	2 088	395	1693
Rotten apples	2 943	601	2342
Fresh oranges	1 854	388	1466
Rotten oranges	1 998	403	1595
Fresh bananas	1 962	381	1581
Rotten bananas	2 754	530	2224
Total	13 599	2 698	10 901

4.2. Displaying and Loading Dataset

After importing all my modules, I went on to use the matplotlib plot function to display individual fruit from the annotated classes to make sure the images display and to test the annotations. Below the script is displayed for fruits from each of the 6 classes while Figure 4.1 shows the displayed images.

```
drive.mount('/content/drive')

PATH = '/content/drive/My Drive/Rotten fruits dataset/dataset'
train_dir = os.path.join(PATH, 'train')

BATCH_SIZE = 32
IMG_SIZE = (160, 160)
```

```
train_dataset = image_dataset_from_directory(train_dir,
                                             shuffle=True,
                                             batch_size=BATCH_SIZE,
                                             image_size=IMG_SIZE,
                                             label_mode='categorical')
class_names = train_dataset.class_names

plt.figure(figsize=(10, 10))
for images, labels in train_dataset.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

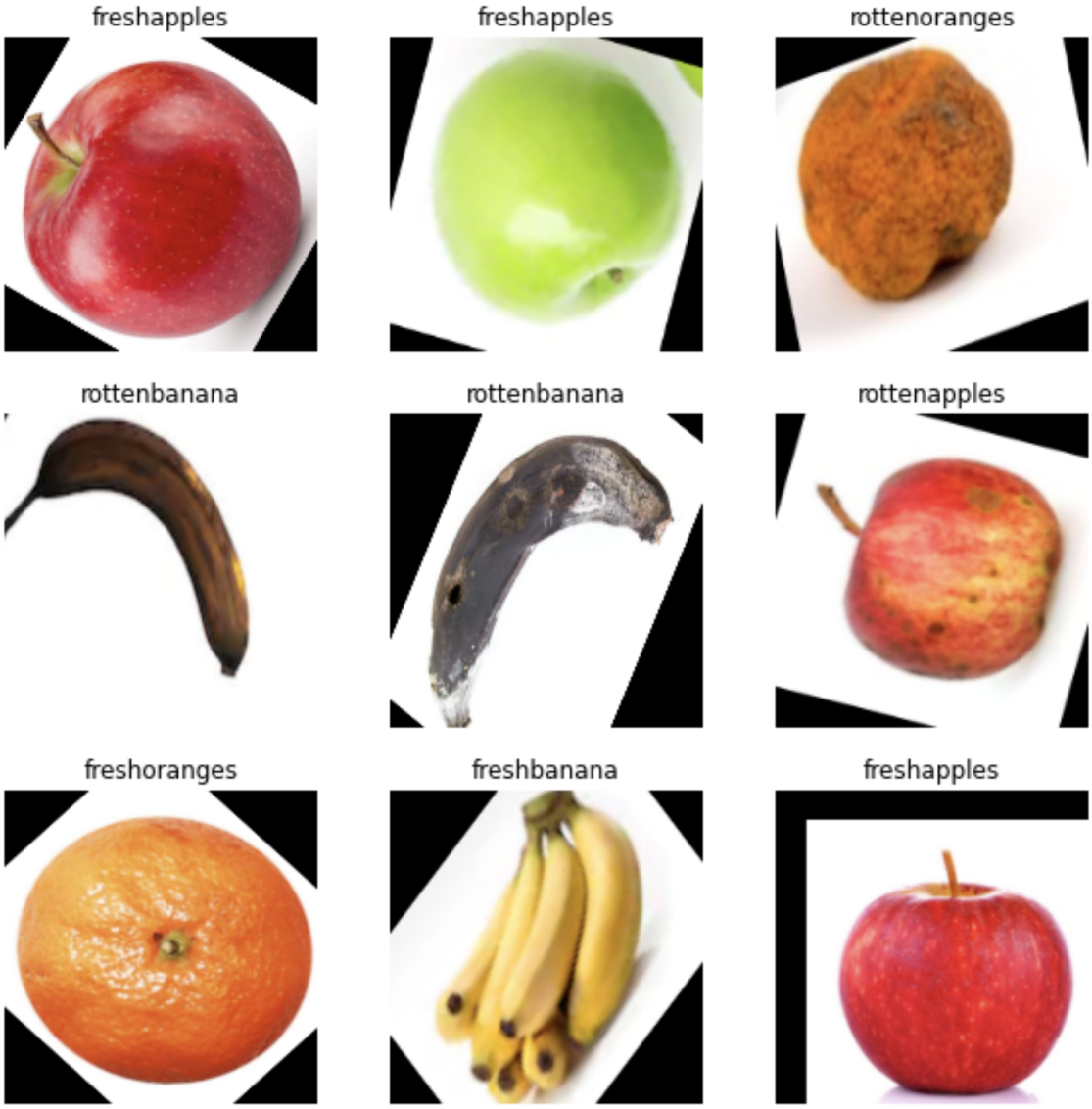

Figure 4.1. Displayed images for fruits

4.3. Building the Model

Here, I designed the model based on my determined CNN architecture. I simply have a convolutional layer, global average pooling, and an output layer with a SoftMax activation function. The code below shows the script for building the CNN model:

```
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
base_model.trainable = False

image_batch, label_batch = next(iter(train_dataset))
feature_batch = base_model(image_batch)
print(feature_batch.shape)

base_model.summary()

global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
feature_batch_average = global_average_layer(feature_batch)
print(feature_batch_average.shape)

prediction_layer = tf.keras.layers.Dense(6, activation='softmax')
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)

inputs = tf.keras.Input(shape=(160, 160, 3))
x = preprocess_input(inputs)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

4.4. Compiling and Training the Model

After building the CNN model, the next step was to compile and train the model on the dataset. The code below shows the script for compiling the model:

```
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_rate
),

loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

The code below shows the script for training the model:

```
history = model.fit(train_dataset,
                    epochs=EPOCHS,
                    validation_data=validation_dataset)
```

5. Evaluation and results

The system was compiled and trained properly. I used 10,901 images for the training of the network over 20 epochs and it took about 300 seconds for every epoch to train which is reasonable considering the size of the dataset and the complexity of the neural network. I tried to classify images for fruits from the 6 classes and it took about 0.387 seconds in average time for classifying a single image (running the classification 10 times and taking its average). For testing, in total, we have 2,698 fruit images.

I calculated my accuracy for training and testing with formulae as shown in the scripts below.

Example of calculating Testing accuracy:

```
loss_final, accuracy_final = model.evaluate(test_dataset)

print("Final loss: {:.2f}".format(loss_final))
print("Final accuracy: {:.2f}".format(accuracy_final))
```

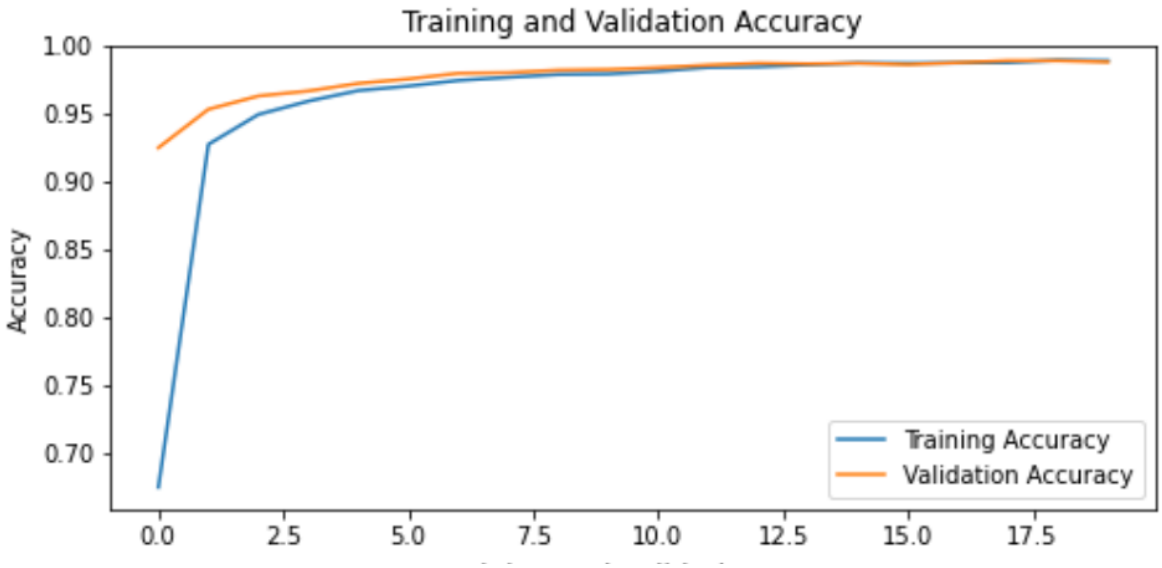

Figure 5.1. Plotting the learning curves

Once the model has been trained, users can upload a picture and the model will give a prediction of its freshness. Depending on the freshness, the model will print “fresh”, or “rotten” respectively.

To overcome the limitation on dataset, we could explore using images from user’s camera devices to improve and/or validate our model. For example, once a user used the application to predict the freshness of the fruit, we could prompt the user to validate whether our prediction was correct.

Using MobileNetV2, I was able to obtain final accuracy of 0.99 and loss of 1.06 on the test examples and 20 epochs was used to train the model. To further test the model, I selected a few test inputs from Google images. To make sure the model could be applied to other types of fruits, I tried to select fruits that would have looked similar during different levels of freshness.

6. Server-side implementation

To implement the server side, 2 endpoints were created:

- endpoint with GET request - used to display the main page with a menu where the user can upload a picture or turn on the camera and take a photo;
- endpoint with POST request - used to upload an image to the server and evaluate its class based on a trained model.

The script of the server-side implementation is defined below:

```
app = Flask(__name__)
CORS(app)

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}
app.config['MAX_CONTENT_LENGTH'] = 2 * 1024 * 1024 # 2 MB

MODEL_PATH = os.path.join(
    'config',
    'fruit_classifier_v3_transfer_learning.h5'
)
model = FruitNet(MODEL_PATH)


def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/', methods=['GET'])
def main():
    return render_template('index.html')


@app.route('/', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({'message': 'No image found.'}), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify({'message': 'Image wasn\'t selected.'}), 400
    if file and allowed_file(file.filename):
        label, percentage = model.predict(file.read())
        return jsonify({ 'label': label, 'percentage': percentage })
    return jsonify({'message': 'Only JPEG or PNG images are
allowed.'}), 400
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

7. Conclusion

From the project, we can conclude that there are limits to the use of transfer learning. While transfer learning using predetermined weights in the CNN can help to speed up the training process, we must be careful in selecting fruits that have similar visual properties with one another to avoid misclassification by the model.

In addition, the small dataset used to train the freshness model may not have provided the model with enough information to distinguish between different fruits with similar visual properties.

Hence, if a larger dataset with more images of similar fruits is used to train the model, the model might be able to correctly classify the fruits. Image recognition performance while at the same time reduce the number of calculations needed to execute the model.

The system developed in this project takes advantage of the computer vision capabilities of convolutional neural networks to help in fruit classification and its freshness, however, the system can still be improved to do even more.