ITRI 626 - Individual Project

Deep Learning for Fruit Type and Freshness Classification Using CNNs

Therese Taylor – 25830473 28 October 2024

1. Introduction

The purpose of this project is to develop a CNN model that can accurately classify different fruit types and their freshness levels. This automated classification has many applications, for example in agriculture farmers and packing facilities can use it to automatically sort freshly harvested fruits based on ripeness, ensuring only high-quality produce is sent to market. For instance, the system could classify and prioritize "ripe" or "nearly ripe" fruits for immediate distribution while separating less mature or overripe produce for other uses, like juicing or composting.

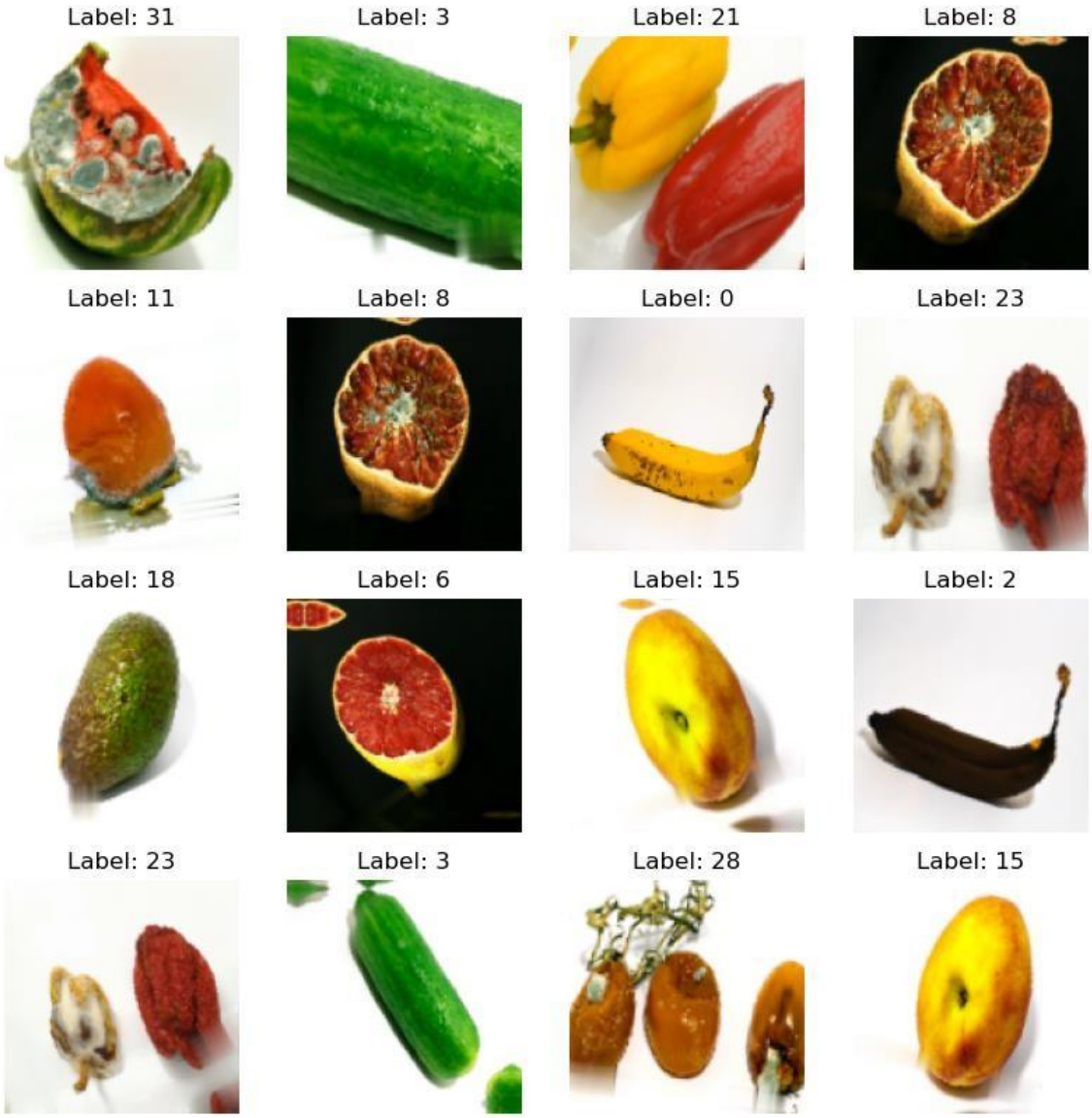
2. Data collection and preprocessing

The dataset used for this project is the FruQ dataset, which includes 11 fruit types: Banana, Cucumber, Grapefruit, Kaki, Papaya, Peach, Pear, Pepper, Strawberry, Tomato, and Watermelon. Each fruit is categorized into three freshness levels: Good, Mild, and Rotten, except for strawberries, which only have Mild and Rotten categories. The data is loaded from folders, and each class is assigned a unique label, such as "Banana_Good," resulting in a total of 32 classes. The dataset consists of 9,370 images, with the number of images per class varying between 24 and 950.

The dataset was split into three parts: 80% for training, 10% for validation, and 10% for testing. The training set allows the model to learn patterns within the data, while the validation set, used during training, helps fine-tune the model and prevents overfitting by providing feedback on performance with unseen data. Finally, the test set evaluates the model's accuracy on new data, ensuring that the model generalizes well to real-world applications.

To enhance generalizability, various augmentation techniques, including flipping, rotation, and zoom, were applied to the training set. All images were also resized and normalized. Specifically, each image was resized to 128x128 pixels to maintain consistent input dimensions, and pixel values were scaled to the range [0, 1] by dividing by 255, which reduces input variance. Additionally, the data was organized into batches of 32 images for efficient processing during training, allowing the model to update weights based on multiple samples at once.

Sample testing set images after augmentation:



3. Model architecture

A CNN model was chosen as it is ideal for image classification because it can automatically learn to recognize patterns like edges, shapes, and textures in images. A custom CNN was created in this project with the following layers and configuration:

- Model configuration:
    - A Conv2D layer with 32 filters of size (3, 3) and ReLU activation that scans the input (128x128 RGB images) for features like edges and textures. ○ A MaxPooling2D layer reduces the spatial size by retaining the highest values in each 2x2 window.

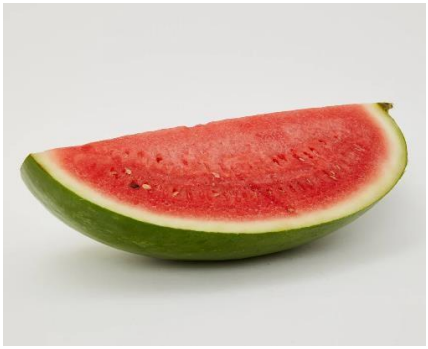
- The Flatten layer transforms the 2D data into a 1D vector, allowing it to be processed by fully connected layers.
- The first Dense layer with 128 neurons and ReLU activation learns complex patterns based on features extracted by the convolutional layers. ○ Finally, an output Dense layer with len(label_map) neurons (total number of classes) and softmax activation produces a probability distribution across the classes, enabling multi-class classification.

- Model compilation: the model is compiled with the Adam optimizer for efficient training, using sparse categorical crossentropy as the loss function to handle integerlabeled classes, and accuracy as the evaluation metric.
- Callbacks: EarlyStopping monitors validation loss, stopping training if there's no improvement over three epochs (helps prevent overfitting).
- Training: The model is trained with fit, using the training dataset, validating with val_dataset, and running for 10 epochs.

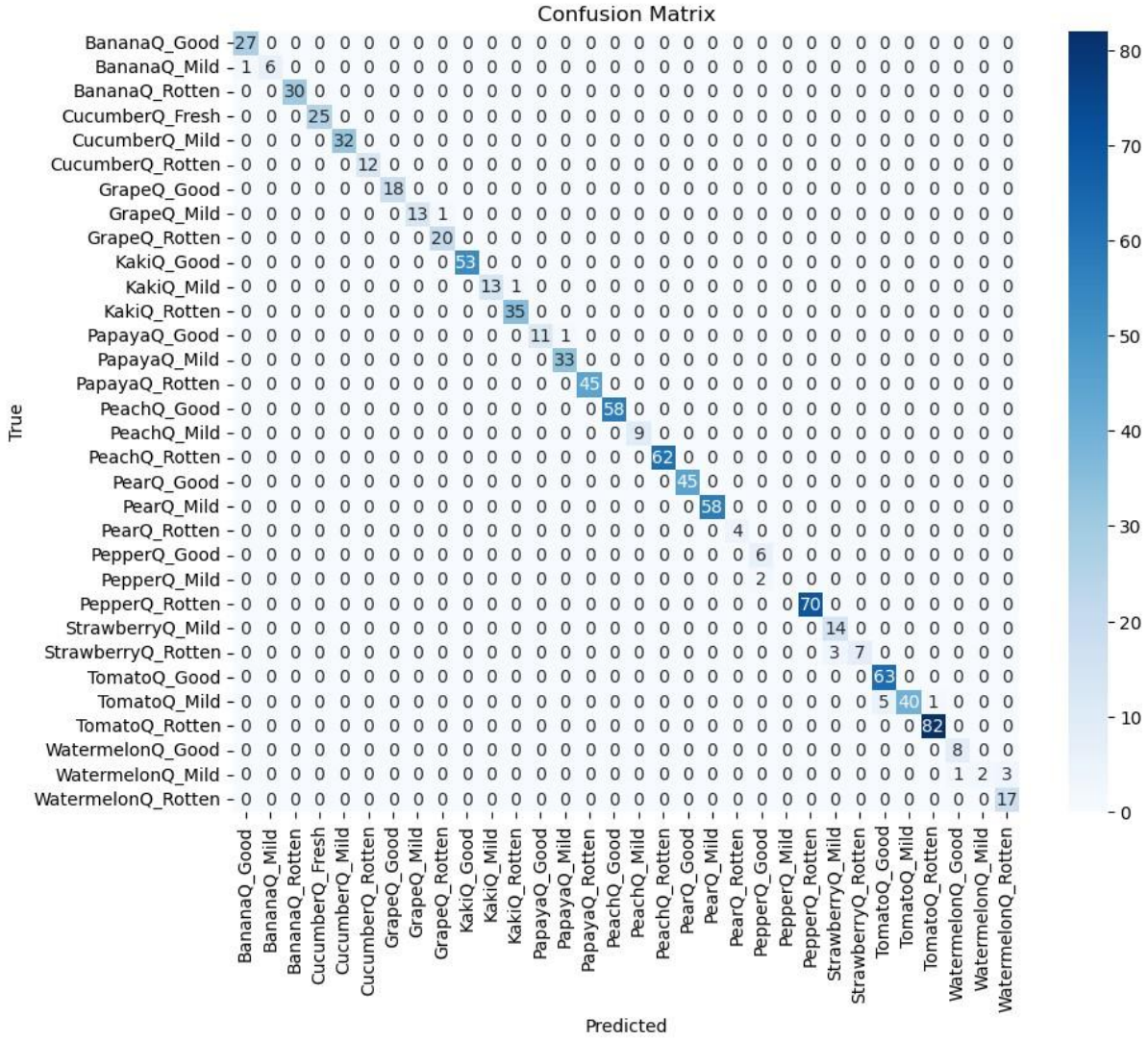
4. Model performance

The model can be validated based on these evaluation metrics:

- *Accuracy*: 98% (testing accuracy).
- *Precision:* 94%
- *Recall:* 93%
- *F1-score:* 92%
- *Generalization:* The model initially performed poorly before data augmentation was applied. After augmentation, its performance improved, but it still lacks strong generalizability. For instance, prior to augmentation, the model misclassified a new image of a watermelon as a grapefruit; following augmentation, it now identifies it as a watermelon, though incorrectly classifying it as a rotten one.

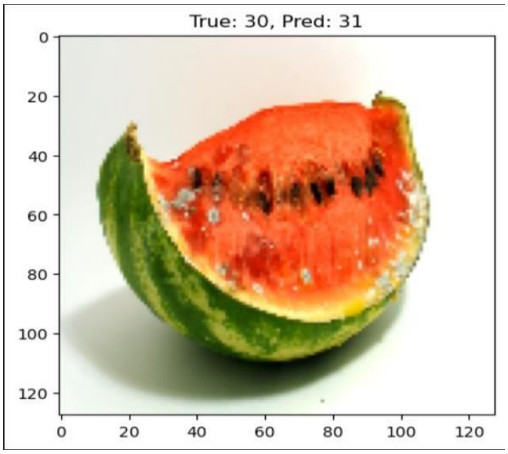


- *Confusion matrix:* The matrix shows that there are just a few misclassifications.



The ones identified incorrectly are mostly within the correct class, just the wrong category.

For example, this is classified as Rotten, but should be Mild:



The model performed well on this dataset; however, due to the dataset's clean images, it struggles with generalizability. To address this, training on a more diverse dataset or using transfer learning with a pre-trained model could improve its ability to handle more varied and real-world images, enhancing overall performance. Classes with fewer images, like the peppers class, also show lower accuracy. This could also explain the difference between the accuracy score and the other metrics. This could be improved by increasing the number of images for these underrepresented classes.