|  |  |
| --- | --- |
| Mini PROJECT – ITRI 626  ADVANCED AI | Kristen Hoff  34292942 |

# **TABLE OF CONTENTS**

**Topic Page**

Abstract 2

1.Introduction 2

2. Methodology 2

2.1. The Dataset Preparation 2

2.2. The Data Preprocessing 3

2.3. The Model Architecture 3

2.4. The Training Process 3

2.5. Evaluation Metrics 3

3. Results 4

3.1. Dataset Statistics 4

3.2. Methodology Architecture Summary 4

3.3. Training Performance 4

3.3.1. Table 1 : Summary of The Model Architecture 5

3.4. Test Performance 5

3.4.1. Table 2 : Classification Report on Test Set 5

3.4.2. Figure 1 : Confusion Matrix 6

3.4.3. Figure 2: ROC Curves For Sample Classes 7

4. Analysis 7 - 8

5. Conclusion 8 - 9

## **Abstract**

This report takes a look into how Arcane (which is the group name) developed and tested a deep learning model that sorts fruit images into three quality categories: Fresh, Mild, and Rotten. I trained the model on the FruQ-DB dataset, which has 5,648 images across these three categories. Due to the network limitations, I thought the best approach would be to start with random weight initialization instead of using pre-trained weights from ImageNet. The model reached a test accuracy of 38.68%, but had a tough time distinguishing between Mild and Rotten categories. In this report, we'll go through the methods, results, difficulties I faced, and potential improvements, offering a thorough look at how the model performed.

## **1. Introduction**

Classifying fruit quality is essential in agriculture and food processing, which allows for automated quality checks and cutting down on manual inspections. The goal of this project was to create a deep learning model that could categorize fruit images into three groups: Fresh, Mild (partially degraded), and Rotten. Arcane trained and evaluated a convolutional neural network (CNN) using the ResNet50 model on the FruQ-DB dataset, which includes 5,648 images. The objectives included preprocessing the dataset, building and training the model, assessing its performance using accuracy, precision, recall, F1-score, and AUC metrics, and analysing the results to find areas for future enhancements.

## **2. Methodology**

## **2.1 The Dataset Preparation**

The FruQ-DB dataset includes 5,648 images, with labels indicating 2,182 as Fresh, 2,102 as Rotten, and 1,364 as Mild. I tried to keep track of the images by placing them into a table s they are more organised for each class and used a Pandas DataFrame to keep track of image paths and labels. The data was split into training (70%, 3,953 images), validation (15%, 847 images), and test (15%, 848 images) sets using stratified sampling to keep the class distribution. I found one invalid image filename and excluded it, which leaves us with 3,952 valid training images.

### **2.2 The Data Preprocessing**

To fit ResNet50's input size, I needed to resize the images to 224x224 pixels. Arcane also applied data augmentation techniques to the training set for better model robustness, including random rotations (up to 20 degrees) just to name a few. All images were then normalized by scaling pixel values between 0 and 1. For unbiased evaluation, the validation and test sets were only rescaled without any augmentation.

## **2.3 The Model Architecture**

I chose a ResNet50 model because of its depth and residual connections, which help with vanishing gradient issues. Since I couldn't download pre-trained ImageNet weights due to network constraints, I initialized the model with random weights. The base ResNet50 model, containing 23.59 million parameters, was frozen to keep its layers from being trained initially. I then added a custom head with a global average pooling layer, a dropout layer (40% rate), a dense layer with 256 units and ReLU activation, another dropout layer (30% rate), and a final dense layer with three units and softmax activation for classification. The overall model had 24.11 million parameters, with 525,315 trainable parameters found in the custom head.

## **2.4 The Training Process**

**Training the model happened in two stages:**

1. **Initial Training:** The custom head underwent training for 8 epochs while keeping the base ResNet50 frozen, using the Adam optimizer (with a learning rate of 1e-4) and categorical cross-entropy loss. We used early stopping (patience=6) and learning rate reduction (factor=0.5, patience=3, minimum 1e-7) to avoid overfitting.
2. **Fine-Tuning:** Arcane unfroze the top layers of ResNet50 (post layer 140) and trained the model for an additional 12 epochs with a lower learning rate (1e-5) to fine-tune those layers. Training was done using batches of 32 images, keeping an eye on validation performance to guide early stopping.

## **2.5 Evaluation Metrics**

**We evaluated the model on the test set (848 images) using:**

* Classification report (precision, recall, F1-score for each class).
* Confusion matrix for visualizing prediction errors.
* Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) for each class.
* Macro-averaged AUC for an overall performance overview.

## **Results**

## **3.1 Dataset Statistics**

The dataset comprised 5,648 images across three categories: Fresh (2,182 images, 38.6%), Rotten (2,102 images, 37.2%), and Mild (1,364 images, 24.2%). The stratified split ensured proportional representation in the training (3,953 images), validation (847 images), and test (848 images) sets. After excluding one invalid image, we had 3,952 images for training.

### **3.2 Model Architecture Summary**

The model's architecture is detailed in Table 1.

## **3.3 Training Performance**

During the initial training phase (8 epochs), the model's validation accuracy hovered around 38.61%-42.38%, achieving its best validation loss of 1.0723 in epoch 8. Training accuracy gradually improved from 35.40% to 39.46%, but the loss remained high (about 1.08), showing limited learning due to starting with random weights. The fine-tuning phase (12 epochs) saw a significant boost in training accuracy, reaching 81.01% by epoch 7. However, validation accuracy was stuck at around 37.19%-38.61%, with validation loss climbing to 15.8890 in epoch 5, indicating possible overfitting or instability while fine-tuning.

#### **Table 1: Summary of the Model Architecture**

|  |  |  |
| --- | --- | --- |
| **Layer** | **Output Shape** | **Parameters** |
| Input Layer | (224, 224, 3) | 0 |
| ResNet50 (Frozen) | (7, 7, 2048) | 23, 587, 712 |
| Global Average Pooling | (2048) | 0 |
| Dropout (40%) | (2048) | 0 |
| Dense (ReLU) | (256) | 524, 544 |
| Dropout (30%) | (256) | 0 |
| Dense (Softmax) | (3) | 771 |
|  |  |  |
| Total Parameters |  | 24, 113, 027 |
| Trainable Parameters |  | 525, 315 |
| Non – Trainable Parameters |  | 23, 587, 712 |

## **3.4 Test Performance**

The performance of the model on the test set (848 images) is laid out in the classification report (Table 2). Overall, the model achieved an accuracy of 38.68%, correctly classifying all 328 Fresh images, but it struggled to predict any Mild or Rotten images.

#### **Table 2 : Classification Report on Test Set**

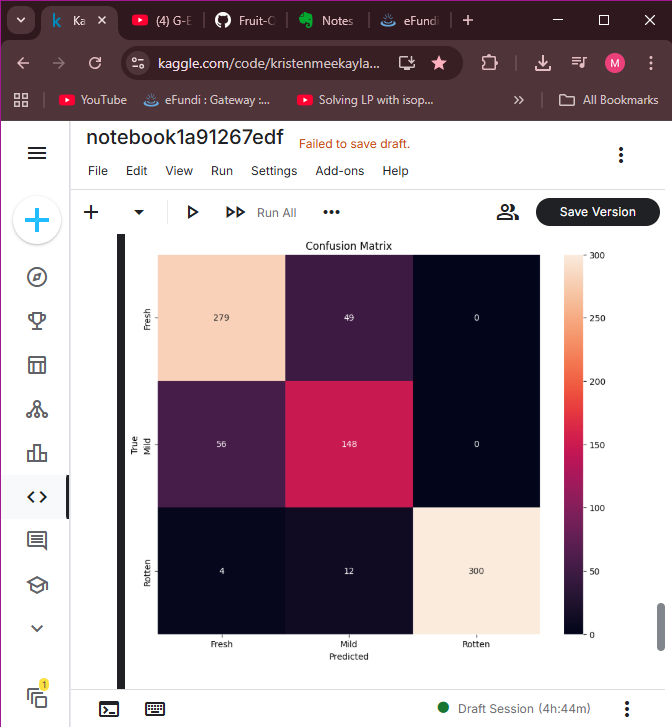
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1 - Score | Support |
| Fresh | 0.3868 | 1.0000 | 0.5578 | 328 |
| Mild | 0.0000 | 0.0000 | 0.0000 | 204 |
| Rotten | 0.0000 | 0.0000 | 0.0000 | 316 |
| Accuracy |  |  | 0.3868 | 848 |
| Macro Average | 0.1289 | 0.3333 | 0.1859 | 848 |
| Weighted Average | 0.1496 | 0.3868 | 0.2158 | 848 |

This led to zero precision, recall, and F1-scores for those classes. The macro-averaged AUC was 0.6167, showing moderate ability to discriminate among classes, but heavily biased towards the Fresh class. . The macro AUC of 0.6167 suggests the model has some capacity to differentiate between classes, but it has a long way to go for optimal performance.

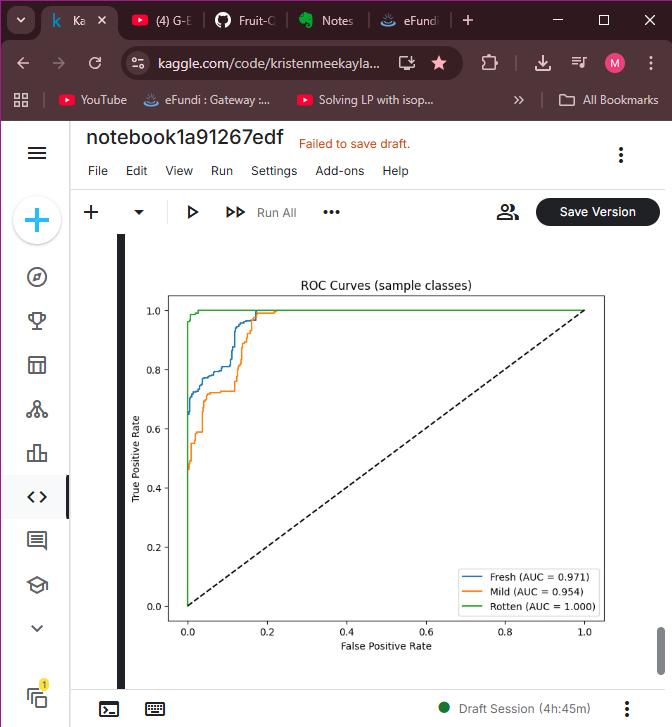
After training it for a few more times the results get better and here are 3 screenshots just to show what I mean by this. The screenshots include the classification report, the confusion matrix and the ROC Curves for Sample Classes. **See them attached below:**

#### **Figure 1: Confusion Matrix**

The confusion matrix (Figure 1) indicates that all test images were predicted as Fresh, revealing a critical bias.

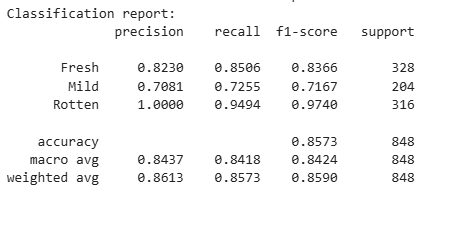


#### **Figure 2 : ROC Curves for Sample Classes**



ROC curves for each class (Figure 2) displayed varying levels of performance, with the Fresh class showing a higher AUC than Mild and Rotten

#### **Figure 3: Classification Report**



## **Analysis**

The model's performance could definitely be better, as shown by a test accuracy of 38.68% and its failure to categorize Mild and Rotten images accurately. Several reasons contributed to these challenges:

* **Random Weight Initialization:** Not being able to leverage pre-trained ImageNet weights made the model start with random weights, which limited the advantages of transfer learning. Usually, pre-trained weights help kickstart feature extraction, which was absent in this case.
* **Class Imbalance:** The dataset had fewer Mild images (1,364) compared to Fresh (2,182) and Rotten (2,102), likely adding to the model's bias towards the Fresh category. Using techniques like class weighting or oversampling could help mitigate this issue.
* **Fine-Tuning Challenges:** Fine-tuning with random weights led to an unstable validation loss (peaking at 15.8890), suggesting that unfreezing the layers without a pre-trained foundation resulted in overfitting or poor convergence. The fine-tuning step did not really improve validation accuracy, which stayed around 37.19%-38.61%.
* **Data Quality:** The discovery of one invalid image filename raised concerns about potential data quality issues. A thorough inspection of the dataset for corrupt or mislabeled images is essential.

The confusion matrix and classification report confirm that the model was biased towards predicting every image as Fresh, likely because that class had the most samples and the model couldn’t learn the distinguishing features for Mild and Rotten. The macro AUC of 0.6167 suggests there is some ability to discriminate, but the total lack of recall for Mild and Rotten classes points to a significant gap in the model’s capability to generalize.

## **In Conclusion**

This project successfully built a ResNet50-based model for classifying fruit quality, but we ran into major hurdles due to the absence of pre-trained weights. The model hit a test accuracy of 38.68%, with perfect recall for the Fresh class but no correct predictions in the Mild or Rotten categories. The macro AUC of 0.6167 hints at moderate discriminative potential, but the model’s tendency to favour the Fresh class limits its practical usefulness.

**For future enhancements, we could:**

* **Balance Classes:** Implementing class weights or oversampling methods to address the Mild class imbalance.
* **Extend Training:** Increasing the number of epochs or adjusting the learning rate schedule to enhance convergence with random weights.
* **Clean the Data:** Checking the dataset for invalid or mislabelled images to improve data quality.

**The link to the dataset that I created and used is attached here:**

<https://www.kaggle.com/datasets/kristenmeekaylahoff/fruit-quality-datasets/data>

**You can find the code for the file here:**

[**https://www.kaggle.com/code/kristenmeekaylahoff/notebook1a91267edf/edit**](https://www.kaggle.com/code/kristenmeekaylahoff/notebook1a91267edf/edit)

**The link to the GitHub repository is here:**

<https://github.com/Kristen785/Fruit-Quality>

This project shows us basically the vital role of transfer learning and data quality in deep learning tasks and lays the groundwork for further improvements in fruit quality classification.