Model Development Phase

Date	20 June 2025
Team ID	SWTID1750180744
Project Title	Smart Sorting: Transfer Learning For Identifying Rotten Fruits And Vegetables
Maximum Marks	5 Marks

Model Selection Report

This report presents the in-depth analysis and rationale behind the selection of the VGG16 model for classifying fruits and vegetables as healthy or rotten. Although VGG16 was the only model implemented, ResNet50 and a custom CNN were also considered during the planning phase. The sections below provide detailed explanations on the capabilities, suitability, and implementation considerations of each model in the context of the project.

Model Selection Report:

Model	Description
Model 1: **VGG16** (Implemented and Trained)	VGG16 is a widely recognized convolutional neural network (CNN) that has shown remarkable performance in image classification tasks. Pre-trained on the ImageNet dataset, it has learned a vast number of visual features across thousands of object categories. This project utilized VGG16 through transfer learning, a strategy that allows leveraging pre-trained models for new but similar tasks. The convolutional base of VGG16 was frozen, meaning its weights were preserved to retain previously learned features, while new dense layers were added for binary classification between healthy and rotten images. One of the primary reasons for choosing VGG16 was its structured and straightforward architecture. It stacks small 3x3 convolutional filters and applies max pooling in between, allowing it to efficiently learn spatial hierarchies. This architecture makes VGG16 not only powerful but also

	interpretable and manageable in terms of computational resources compared to more complex models like ResNet50. For this project, the VGG16 model required relatively minimal fine-tuning and provided consistent convergence behavior during training. Furthermore, the model's performance on the validation dataset was particularly strong, achieving around 88% accuracy. This level of performance is noteworthy given that the dataset was limited in size and had moderate class imbalance. The use of ImageDataGenerator for augmentation also helped mitigate overfitting, allowing the model to generalize well to unseen data. Training time was efficient, and the GPU memory usage remained well within resource constraints. In the practical implementation of the project, VGG16 was utilized as the primary model for image classification. The process began with importing the pre-trained VGG16 model using TensorFlow's `keras.applications` module. The model was configured with `include_top=False` to remove the default classification layers, keeping only the convolutional base. This base was frozen to retain the high-quality feature extraction it had learned from ImageNet. To adapt VGG16 to the specific task of binary classification (healthy vs. rotten), a new classification head was added. This included a Flatten layer to convert the output into a vector, followed by a Dense layer with ReLU activation, and a final Dense layer with softmax activation to output class probabilities. The model was compiled using the Adam optimizer and trained using categorical cross-entropy loss.
Model 2: ResNet50 **(Considered but Not Implemented)**	ResNet50, a deep residual network with 50 layers, was among the models considered for this project. Designed to solve the degradation problem that arises when increasing the depth of neural networks, ResNet50 introduced the concept of residual learning with skip connections. These shortcuts allow gradients to flow more effectively through deep networks, enabling more accurate training of deeper architectures. ResNet50 is known for its excellent performance in large-scale image recognition tasks and has been widely used in academic and industrial settings. During the early planning phase of this project, ResNet50 was considered a promising alternative due to its ability to capture more complex patterns. The model was imported via TensorFlow's `tensorflow.keras.applications` module, and its potential integration was explored. However, the practical

	implementation was ultimately set aside in favor of VGG16. There were multiple reasons behind this decision. First, the computational requirements for training ResNet50 are considerably higher. It demands more GPU memory and takes longer to process each epoch. Given the resource limitations for this project, running and fine-tuning ResNet50 would have required compromises in batch size or input resolution, possibly affecting model performance. Second, while ResNet50 may perform better on very large and complex datasets, our dataset—although diverse—was relatively small and well-suited for a shallower model like VGG16. VGG16 had already begun to show strong validation results with minimal tuning, reducing the incentive to switch to a more demanding model. Furthermore, VGG16 provided simpler integration and better interpretability. While ResNet50 excels in many domains, the trade-offs in training time and complexity did not justify its use in the context of this project. The team chose to prioritize stability, efficiency, and repeatability—attributes for which VGG16 was better suited.
Model 3: Custom CNN **(Explored Conceptually, Not Developed)**	A custom Convolutional Neural Network (CNN) was also conceptually considered during the project planning stage. Building a CNN from scratch provides the advantage of full control over model architecture, hyperparameters, and training strategy. A typical custom CNN would include several Conv2D layers followed by MaxPooling2D layers, with activation functions like ReLU and final classification via dense layers. The idea was to create a lightweight model specifically tailored to the binary classification task of distinguishing between healthy and rotten produce. This model would offer high training speed and low memory consumption. However, after further assessment, this approach was deprioritized for several reasons. Firstly, training a custom CNN from scratch requires a substantial amount of data to ensure that the model can generalize effectively. Given the size of our dataset, the risk of overfitting a shallow custom model was high. Second, transfer learning provides a powerful shortcut by leveraging pre-trained feature extractors like those in VGG16, which have been trained on millions of images. In comparison, a custom CNN would have started with random weights, likely resulting in lower accuracy unless heavily regularized and fine-tuned.

	Third, the time and effort needed to trial multiple custom architectures was not feasible within the timeline of the project. Instead, a proven pre-trained model like VGG16 offered a more reliable and time-efficient pathway. The goal was to achieve high accuracy and strong generalization without unnecessary experimentation. Ultimately, although the custom CNN approach was explored conceptually, it was not developed or tested. The decision was driven by practical considerations related to data volume, project scope, and the availability of pre-trained alternatives. The team focused resources on optimizing the VGG16-based solution, which delivered excellent performance with relatively minimal tuning.