Tomato Quality Classification Based on Transfer Learning Feature Extraction and Machine Learning Algorithm Classifier

Aswani V A[1], Bency S[2], Christo Sabu[3], Gopika S[4], Mrs. Vidya C A[5]
Rajadhani Institute of Engineering & Technology

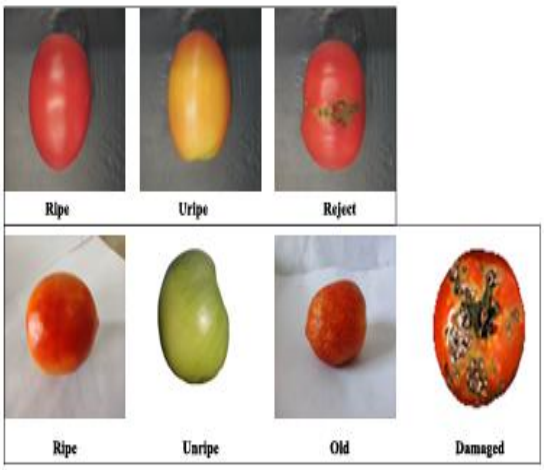
***Abstract*—The System propose an automated tomato quality classification system using Support Vector Machine (SVM) for classification and Inception V3 as a feature extractor. Traditional methods for assessing tomato quality rely on manual inspection, which is time-consuming, inconsistent, and prone to human error. To address these challenges, we leverage deep learning-based feature extraction and machine learning classification for accurate and efficient quality assessment. Inception V3, a pre-trained Convolutional Neural Network (CNN), is utilized to extract high-level features from tomato images, capturing essential texture, color, and shape attributes. These extracted features are then passed to an SVM classifier, which categorizes tomatoes based on predefined quality classes such as ripe, unripe, and defective. The model is implemented using Python, ensuring scalability and ease of deployment. The proposed system enhances accuracy, consistency, and automation in tomato quality classification, making it suitable for agricultural and food processing industries. Experimental results demonstrate that the Inception V3 + SVM combination improves classification performance compared to traditional feature extraction and machine learning techniques.**

***Index Terms*—Convolution Neural Network, Support Vector Machine, InceptionV3.**

I. INTRODUCTION

The tomato is one of the most common and familiar vegetables in our daily lives and is an essential horticultural crop worldwide. Playing a significant role in the food industry, tomatoes are integral to various cuisines and food products across regions. According to FAOSTAT and TILASTO, worldwide tomato production in 2021 reached an impressive 189 million tons. Sorting and grading this vast quantity of tomatoes is, however, a challenging and labour-intensive task. Tomatoes have a range of quality parameters based on features like size, shape, colour, ripeness, and defects. Therefore, the grading process is crucial to ensure that the products meet quality and safety standards for consumers. The traditional method of grading tomatoes relies on a manual approach by trained experts. Each tomato is individually inspected and sorted, a process that is time-consuming and requires substantial labour. Additionally, manual grading can lead to inconsistencies as it varies between experts and is susceptible to human error, fatigue, and subjectivity. This method is not efficient for handling large volumes of tomatoes, making it expensive and difficult to scale. Consequently, the search for alternative, automated methods of sorting and grading tomatoes has gained interest. Adopting such alternatives could enhance the efficiency and accuracy of the grading process. International standards established by organizations like the Organization for Economic Cooperation and Development (OECD) and the United States Department of Agriculture (USDA) help regulate tomato quality for consumption. These standards define criteria such as size, colour, shape, ripeness, and allowable defects, impacting both domestic and international trade. The objective of these standards is to ensure fair and transparent trading practices across member countries. Common tomato defects include surface cracks, scars, sunburn, insect damage, discoloration, irregular marks, distortions, bruises, and blossom end rot. Adhering to these standards is vital for growers, distributors, and consumers to maintain quality and meet market demands. Machine learning has proven to be highly effective in diverse applications, such as quality assessment, crop disease detection, and natural language processing, due to its ability to Automate and

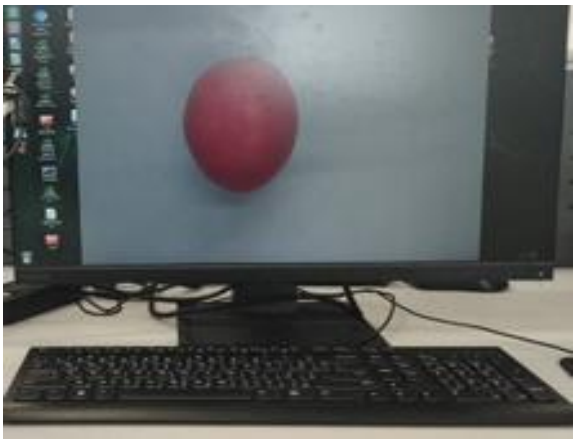
enhance complex tasks. Recently, the food industry has embraced machine learning to address challenges like human error, subjectivity, labour costs, and time inefficiencies in tasks like sorting and grading produce. In tomato quality analysis, various machine learning

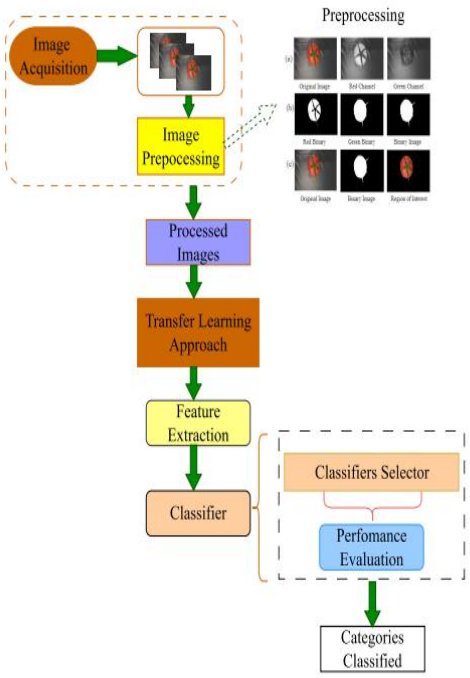


approaches have been developed to accurately assess tomatoes based on key quality parameters. These automated methods aim to improve consistency, reduce labour demands, and increase throughput, making them valuable in large-scale food processing environments. Many image processing techniques, including colour-based segmentation, morphological operations, and edge detection, are now integral to automated tomato grading. These techniques enable machine learning models to focus on visual features that determine quality, such as colour and shape. Furthermore, algorithms like Convolutional Neural Networks (CNN), Support Vector Machines (SVM), k-Nearest Neighbours (KNN), and Random Forests (RF) are commonly used to classify tomatoes based on these visual cues. While machine learning and deep learning offer significant advantages, they also introduce challenges, such as increased system complexity, longer training times, and the need for high computational resources, which can impact the feasibility of deploying these models at scale.

II. MATERIALS AND METHODS

In this section, we present the proposed framework for tomato classification





A.IMAGE ACQUISITION

The images were acquired in an uncontrolled lighting environment. The system operated at 30 frames per second, capturing tomato images one at a time. The sum of 600 tomatoes with different shapes, sizes, colors, and defects collected from the local market was washed and dried. The image acquisition involved capturing four images (four sides of the tomato i.e., top, bottom, rear, and front as shown in Figure 3) from each tomato, resulting in a total of 2400 images.

B. IMAGE PREPROCESSING

The captured tomato images have non-uniform illumination due to the light's reflection, which leads to contrast variation. These variations of the contrast

in the images may impact the performance of the classification model and increase computational complexity. To overcome this problem, we performed segmentation and background cancellation to get the region of interest (ROI) before training our dataset.

C.DATA AUGMENTATION

Data augmentation is a machine learning and computer vision technique to increase the training data available for a model to learn from. It involves applying various transformations to the existing data to create new, slightly altered versions of the original data. Data augmentation aims to improve the generalization and robustness of machine learning models by exposing them to a more extensive and diverse set of examples during training. By introducing variations in the training data, the model can learn to be more f lexible and adaptable and handle new and unseen inputs during inference. We augmented our training set by applying rotation, reflection, translation, and scaling.

D. TRANSFER LEARNING

Transfer learning addresses the challenges associated with training deep learning networks when the amount of available data is limited. Instead of starting from scratch, transfer learning uses pre-trained deep learning networks customized for the specific task. The pre-trained network can be utilized as a feature extractor or for end-to-end classification tasks by carefully adjusting some parameters. This study used a pre-trained network as a feature extractor and classify the extracted features using traditional machine learning classifiers. . We then proceed with fine-tuning the selected pre trained networks. Fine-tuning aims to teach the network to recognize classes that were not trained before. It involves removing the last convolutional and classification layers of the adopted pre-trained network to match the number of classes in the building classification challenge. To prevent overfitting, we froze some network layers, apply batch normalization, and dropout during training.

E. FEATURE EXTRACTION

In our proposed system, the selected pre-trained networks were used to extract the deep features from the last convolutional layers, or dense layers of the network. Usually, these networks are designed with more convolutional layers to increase network performance. A set of weight layers cascade with another layer separated by the activation layer such as a rectified linear unit (ReLU). Thus, features were extracted from the deepest layer of the network and used to train traditional machine learning classifiers. Here we used InceptionV3 for feature extraction.

F. CLASSIFIERS

Machine learning classifiers are algorithms trained on labeled datasets to learn patterns and relationships between input features and corresponding output labels. The goal of a machine learning classifier is to predict the correct label for a new input based on the relationships learned from the training dataset. Machine learning classifiers include decision trees, random forests, logistic regression, support vector machines, k-nearest neighbors, and neural networks. The choice of a machine learning classifier depends on the specific characteristics of the dataset and the desired performance metrics. A labeled dataset is typically divided into training and validation sets to train a machine learning classifier. The training set is used to train the classifier, and the validation set is used to evaluate the classifier's performance on new, unseen data. The classifier's performance is measured using accuracy, precision, recall, specificity, and F1 score metrics. This study uses support vector machine.

SUPPORT VECTOR MACHINE

The support vector machine is highly effective in image classification and regression tasks because it can handle dimensionality issues. This algorithm employs support vectors for determining the coordinates of individual observations. It can produce accurate outcomes even in high dimensional spaces, such as when the number of dimensions exceeds the number of samples.

III.PERFORMANCE MEASURE

The performance was eval uated based on training time, testing time, accuracy, recall, precision, specificity, and F-1 score calculated according to equation (2) to (6).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

$$Recall(R) = \frac{TP}{TP + FN} \quad (3)$$

$$Precision(P) = \frac{TP}{TP + FP} \quad (4)$$

$$Specificity = \frac{TN}{TN + FP} \quad (5)$$

$$F1 - score = 2\left(\frac{RP}{R + P}\right) \quad (6)$$

where TP, TN, FP, and FN represent true positive, true negative, false Positive, and false negative, respectively

IV. IMPLEMENTATION

Here we go with the implementation part of our proposed system

1. Install and Download Dataset: To install and download the dataset, first, install the Kaggle package using! pip install kaggle, which allows access to datasets from Kaggle. Then, download the "tomato-maturity-detection-and-quality-grading" dataset using the command! kaggle datasets download christo/tomato-maturity-detection-and-quality-grading. This ensures that the dataset required for tomato maturity detection and quality grading is available for further analysis.

2. Unzip the Dataset: To unzip the downloaded dataset, use the command! unzip tomato-maturity-detection-and-quality-grading.zip. This extracts the contents of the ZIP file into the current directory, making the dataset accessible for further processing and analysis.

3. Import Required Libraries: To import the required libraries, use the following Python packages: os for handling file and directory operations, numpy for working with arrays and numerical computations, and cv2 for reading and processing images. The tensorflow library is used to import the InceptionV3 model for feature extraction, while sklearn provides tools for training and evaluating an SVM classifier, including train_test_split for data splitting, SVC for classification, and accuracy_score and classification_report for performance evaluation. Additionally, joblib is used to save and load the trained model efficiently.

4. Set Data Directory and Class Names: To set up the data directory and class names, define data_dir as the path to the dataset: "Tomato Maturity Detection and Quality Grading Dataset/Tomato Quality Grading Dataset/Tomato Quality Grading Dataset/Augment Dataset", which contains the images. The classes list includes two categories, "Fresh" and "Rotten," representing the classification labels. To ensure consistency, all images will be resized to img_size = (299, 299) pixels before processing.

5. Load Pre-trained InceptionV3 Model: To load the pre-trained InceptionV3 model, use the command inception_model = InceptionV3(weights='imagenet', include_top=False, pooling='avg'). This model is pre trained on the ImageNet dataset and is used for feature extraction. Setting include_top=False removes the top classification layers, allowing only the feature extraction layers to be used. Additionally, pooling='avg' applies global average pooling to flatten the extracted features, making them suitable for further processing.

6. Function to Load and Preprocess Images: The load_images_and_labels() function is designed to load and preprocess images for classification. It initializes empty lists for images and labels, then iterates through the "Fresh" and "Rotten" class directories. For each image, it constructs the file path, reads the image using OpenCV (cv2.imread()), and prints the image path for reference. If the image is successfully loaded, it is resized to 299x299 pixels for consistency and preprocessed using preprocess_input() to match InceptionV3's input format. The processed images are appended to the images list, while their corresponding labels (0 for "Fresh" and 1 for "Rotten") are stored in labels. Finally, the function returns both images and labels as NumPy arrays for further use in model training.

7. Load Dataset: To load the dataset, the script first prints "Loading dataset..." as an indication. It then calls the load_images_and_labels() function to retrieve the processed images (X) and their corresponding labels (y). Once the images are successfully loaded, it prints the total number of images using print (f"Loaded {X.shape[0]} images."), providing confirmation that the dataset is ready for further processing.

8. Extract Features Using InceptionV3: To extract features using InceptionV3, the script first prints "Extracting features..." as a status update. It then passes the preprocessed images (X) through the pre-trained InceptionV3 model using

inception_model.predict(X), generating feature vectors that capture important image characteristics. Once the extraction is complete, it prints the shape of the extracted features using printf("Extracted features shape: {X_features.shape}"), providing insight into the dimensions of the feature representations.

9. Split Dataset into Training and Test Sets: To split the dataset into training and test sets, the train_test_split() function is used with X_features as input features and y as labels. The data is divided into 80% for training and 20% for testing by setting test_size=0.2, ensuring a balanced evaluation. Additionally, random_state=42 is specified to maintain reproducibility, meaning the split remains consistent across multiple runs

10. Train SVM Classifier: To train the SVM classifier, the script first prints "Training SVM classifier..." as an update. It then creates an SVM model using SVC(kernel='linear', probability=True), where the linear kernel is chosen for efficient classification, and probability=True enables probability estimates for predictions. The model is trained using svm_model.fit(X_train, y_train), which fits the classifier to the extracted features from the training dataset, allowing it to learn patterns for distinguishing between fresh and rotten tomatoes.

11. Evaluate Model: To evaluate the trained SVM model, predictions are made on the test set using svm_model.predict(X_test), which generates predicted labels for the test data. The model's accuracy is then calculated using accuracy_score(y_test, y_pred), and the result is printed in a formatted manner with print(f"Accuracy: {accuracy:.4f}"). Additionally, classification_report(y_test, y_pred, target_names=classes) is used to display detailed performance metrics, including precision, recall, and F1-score for each class, providing a comprehensive assessment of the model's effectiveness in distinguishing between fresh and rotten tomatoes.

12. Save Trained Model: To save the trained SVM model, the joblib.dump() function is used with the command joblib.dump(svm_model, "svm_tomato_quality.pkl"), which stores the model in a file named "svm_tomato_quality.pkl". This allows the model to be reused later without retraining. Once the saving process is complete, a confirmation message "Model saved as svm_tomato_quality.pkl" is printed to indicate that the model has been successfully stored.

13. Function to Predict a New Image: To predict the class of a new tomato image, the predict_tomato(image_path) function is defined. It first reads the image using cv2.imread(image_path), and if the image fails to load, it prints an error message and exits. The image is then resized to 299x299 pixels for consistency and preprocessed using preprocess_input(). To match the expected input shape for InceptionV3, it is expanded into a batch format using np.expand_dims(img, axis=0). The function then extracts features using inception_model.predict(img), which are subsequently fed into the trained SVM model for classification. The predicted class (either "Fresh" or "Rotten") is displayed using print (f"Predicted class: {classes[prediction]}").

14. Test Prediction with an Example: To test the prediction function, an example image is passed to predict_tomato(). The function is called with the command predict_tomato("/content/Tomato Maturity Detection and Quality Grading Dataset/Tomato Quality Grading Dataset/Tomato Quality Augment Dataset/Rotten/python_original_IMG_20230120_221800.jpg_081cf5ce-f30c 48aa-90bf-f18f859b6e49.jpg"), which processes the specified image and classifies it as either "Fresh" or "Rotten." In this example, the predicted class is "Rotten," confirming the functionality of the model in assessing tomato quality.

V. RESULT

Here are the result of our system,

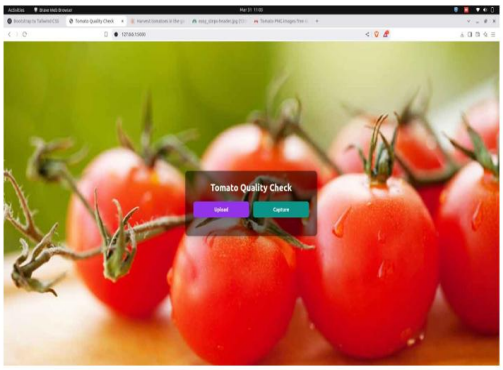


Figure 5.1 represents the home page of Tomato quality classification. The interface features two buttons labeled "Upload" and "Capture" that allow users to either upload an image or capture one using a camera.

Below these buttons, there is a prominent blue button labeled "Upload & Analyze", likely to initiate the analysis process. The page has a clean, minimalistic design with a centered content card and a light gray background.

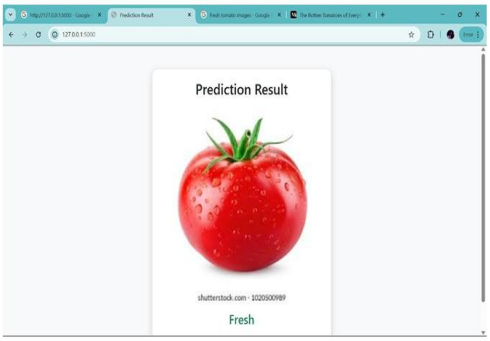


Figure 5.2 shows the "Prediction Result" page of the Tomato Quality Check application, where the system analyzes and determines the quality of the tomato. In this case, the system has classified the tomato as "Fresh," indicating that the tomato is of good quality. The interface displays a clear image of the analyzed tomato, allowing the user to visually confirm the result. Below the prediction result, a button labeled "Upload Another Image" is provided for the user to upload a new image for further analysis. This feature enables users to easily perform multiple checks without restarting the application. The application maintains a user-friendly interface, ensuring a smooth and efficient experience for quality assessment.

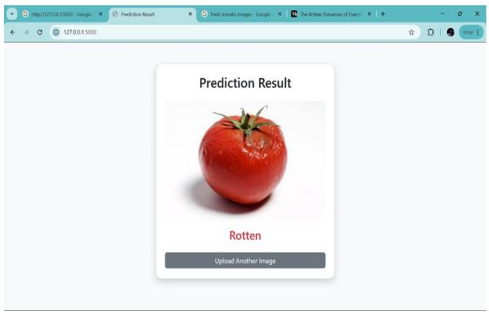


Figure 5.3 presents the "Prediction Result" page of the Tomato Quality Check application, where the system analyzes and classifies the quality of the tomato. In this instance, the system has identified the tomato as "Rotten," highlighting that the tomato is spoiled and no longer suitable for use. The displayed image shows a visibly rotten tomato with noticeable wrinkles and discoloration, supporting the system's prediction. Below the result, the label "Rotten" is highlighted in red, making it clear to the user that the tomato is not fresh. A button labeled "Upload Another Image" is provided at the bottom, allowing the user to analyze another tomato image if desired. This feature ensures that users can easily perform multiple quality checks, enhancing the overall usability of the application.

VI. CONCLUSION

This project proposes an innovative hybrid model that combines convolutional neural network (CNN) feature extraction with machine learning classifiers to achieve efficient and accurate tomato quality grading. By leveraging CNN's deep feature extraction capabilities alongside the precision of machine learning classifiers like support vector machines (SVM), the CNN-SVM model outperformed other traditional methods, showing superior accuracy in both binary and multiclass classification tasks. This model demonstrates remarkable robustness, handling varied real-world conditions such as different lighting and environmental factors, making it highly suitable for practical applications. Additionally, this hybrid approach improves grading accuracy and reduces training complexity and computational costs, offering a more scalable and resource-efficient solution for automated tomato sorting and grading systems. With its successful application in tomato quality assessment, future work may extend this approach to other agricultural crops and explore real-time deployment possibilities, further enhancing its utility in industrial food processing and quality control.

VII. ACKNOWLEDGMENT

We sincerely express our gratitude to everyone who contributed to the development of this Tomato quality classification Based on Transfer Learning Feature Extraction and Machine Learning Algorithm Classifiers. Special thanks to our mentors and advisors for their valuable guidance and technical insights, which played a crucial role in shaping this project. We also appreciate the contributions of our team members for their dedication and collaboration in designing and implementing the System. Additionally, we acknowledge the efforts of researchers and developers

whose work based on this technology. Finally, we extend our thanks to the institutions and organizations that provided resources, support, and encouragement for this project. Their support has been instrumental in developing a reliable, real-time response for users.

REFERENCES

[1] Petridis, Stavros, and Maja Pantic. "Deep multimodal learning for audio-visualspeech recognition." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2794-2803. 2018.

[2] Assael, Yannis M., Brendan Shillingford, Shimon Whiteson, and Nandode Freitas. "LipNet: End-to-end sentence-level lipreading." arXiv preprintarXiv:1611.01599 (2016).

[3] Gan, Chuang, Linlin Chao, and David Cox. "Temporal 3D convnets: New architecture and transfer learning for video classification." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4734-4742. 2018.

[4] S.Stafylakis, Themos, and Gerasimos\Potamianos. "Combining neural networks for audio- visual speech recognition." In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 2371-2375. IEEE, 2018.

[5] Afouras,Triantafyllos,Joon Son Chung, andAndrewZisserman."Deep audio-visual speech recognition." IEEE Transactions on Pattern Analysis and Machine Intelligence 41, no. 5 (2018): 978- 100.

[6] Take off Edu Group Youtube (29 October 2024) on Tomato Quality Classification https://youtu.be/m98CoifwRMw?si=Sfj1NxGrgzJb6a.

[7] J. Amin, M. A. Anjum, R. Zahra, M. I. Sharif, S. Kadry, and L. Sevcik, “Pest Localization using YOLOv5 and classification based on quantum convolutional network,’’ Agriculture, vol. 13, no. 3, p. 662, Mar. 2023.