JOURNAL OF DATA SCIENCE | Vol.2024:44
eISSN:2805-5160

Evaluation and Comparative Analysis of Feature Extraction Methods on Image Data to increase the Accuracy of Classification Algorithms

Rachmad Iqbal [1], Tri Basuki Kurniawan[1*], Misinem[2], Edi Surya Negara[1], Tata Sutabri[1]

[1]Magister of Information Technology, Universitas Bina Darma, Palembang, Indonesia
[2]Faculty of Vocation, Universitas Bina Darma, Palembang, Indonesia

[*]**Email:** tribasukikurniawan@binadarma.ac.id

Abstract

Manual selection of fresh fruit has been identified as a significant challenge for the agricultural sector due to its time-consuming nature and potential for inconsistent categorization. This process requires human labour to visually inspect and sort fruits, leading to variability and inefficiencies in the sorting process. This research proposes a low-cost alternative using intelligent fruit selection systems based on computer vision techniques to address these issues. These systems aim to automate the process of fruit selection, improving efficiency and consistency in categorizing fruits such as apples, bananas, and oranges. A critical step in developing such intelligent systems is the feature extraction process. Feature extraction is essential in classification, especially for data sources in the form of images. It involves identifying and isolating relevant information from the images that classification algorithms can use to distinguish between different fruit categories. If the feature extraction process fails to capture the correct information, the performance or accuracy of the classification algorithm will be negatively impacted. This research compares three different methods for extracting features from fruit images to determine which method yields the highest accuracy for fruit classification. The feature extraction methods evaluated were Grayscale Pixel Values, Mean Pixel Value of Channels, and Extracting Edge Features. The classification algorithm used in this research is the Convolutional Neural Network (CNN) algorithm. CNNs are well-suited for image classification tasks due to their ability to learn hierarchical feature representations from the input images automatically. By comparing the performance of the CNN classifier using the three different feature extraction methods, this research aims to identify the method that provides the highest level of accuracy in classifying fruit images. By conducting this comparative analysis, the research provides insights into the most effective feature extraction techniques for improving the performance of computer vision-based fruit selection systems, ultimately contributing to more efficient and consistent fruit categorization in the agricultural sector. The result revealed that the Grayscale achieved the highest validation accuracy (99.94%) and the lowest validation loss (0.44%).

Keywords

Feature image extraction, Image classification, Convolution neural network (CNN)

Submission: 20 October 2024; **Acceptance:** 21 November 2024

Introduction

The manual selection of fresh fruit has been identified as a significant challenge for the agricultural sector as it is time-consuming and can lead to categorization consistency. Computer vision technology combined with Artificial Intelligence (AI) is suggested in this proposal for the creation of an intelligent fruit selection system to increase the efficiency of the sorting process and realize testing that does not damage the fruit. This research presents a low-cost alternative to intelligent fruit selection systems (apples, bananas, oranges, etc.) using computer vision-based techniques. The intelligent fruit selection system is achieved using several phases of artificial intelligence, such as grayscale processing, binarization, enhancement processing, feature extraction, etc. Following established fresh fruit grading standards, the proposed low-cost vision system is expected to automate fresh fruit selection, lowering labor costs and providing a cost-effective solution for medium- and large-scale businesses. It also includes a detailed analysis of fruit selection using various feature extraction methods.

From several studies have been conducted by several previous researchers, such as those undertaken by Rawat & Wang (2017), Yamashita et al. (2018), and N. Sharma et al. (2018), all three of which used a convolution neural network algorithm in the digital image classification process and obtained excellent accuracy results. Also, in 2022, Ramprasath et al. (2022) used the same method and obtained excellent accuracy results. For this reason, researchers feel it is necessary to choose and use the same process in this research so that the accuracy values that are already good or high can be improved or improved.

Convolutional Neural Networks (Convnets/ CNN) are similar to standard artificial neural networks. These can be visualized as a collection of neurons, nodes, or units arranged as an acyclic graph (a graph without any loops). CNN has the characteristic of having hidden layers only connected to a subset of neurons in the previous layer. Because of this connectivity, CNNs can learn features implicitly.

The CNN architecture produces hierarchical feature extraction, namely filters that are trained for specific purposes; for example, the first layer is usually focused on identifying edges or color fluctuations, then the second layer is usually more focused on identifying shapes, and the next layer of filters is generally more directed at learning partialization parts of the system. Objects, whether seen a little or partially or seen quite a lot, and the last layer is used to identify the object, as shown in Figure 1.

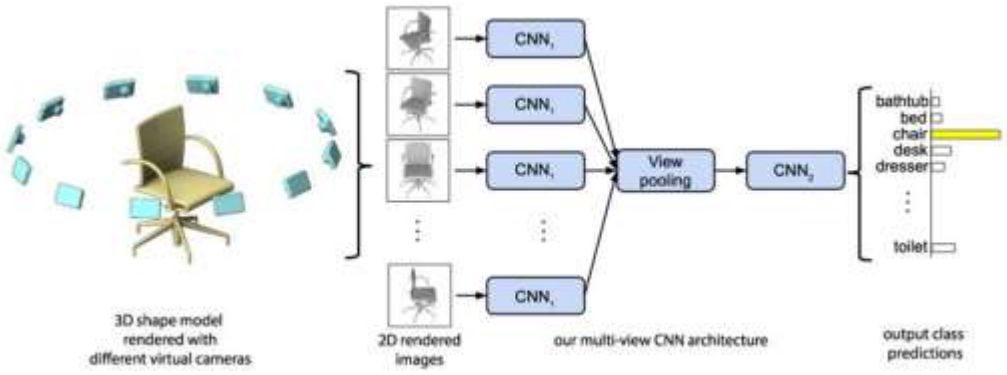


Figure 1 CNN Feature Extraction from 3D to 2D Objects (source: Soebroto, 2019).

The feature extraction process is the most essential part of the classification process. Especially for data sources in the form of images, it is necessary to carry out a careful feature extraction process using computer vision techniques. This is because, if the feature extraction process does not obtain the correct information for the classification training process, the performance or accuracy of the classification algorithm will remain the same.

Information from objects in digital images or image data can be used to identify or differentiate one image from another. The process of retrieving information from the image object is feature extraction, which produces parameters to define objects.

This research field has received much attention from researchers, such as Madalina-Cosmina and Sasu (2014), who compared 11 digital image feature extraction methods in 2014. Next, in 2015, Medjahed (2015) carried out almost the same research but used 14 different types of digital image feature extraction methods.

In recent years, this area of research has still attracted the attention of researchers, as evidenced by research conducted by Sharma and Abrol (2020) and Xu et al. (2023). Each researcher compared different extraction methods. New ideas are always discovered in the extraction process, resulting in higher and better accuracy values in the classification process. For this reason, researchers try to compare the basic methods of extracting digital image features to understand the process better.

In this research, 3 (three) methods of extracting features from data in the form of fruit images will be compared. The feature extraction methods used are Grayscale Pixel Values, Mean Pixel Value of Channels, and Extracting Edge Features.

Methodology Research

This section outlines the comprehensive approach to developing and evaluating the fruit image classification model. The methodology consists of multiple stages, including data collection and classifier processing. Each step is crucial to ensure the accuracy and robustness of the final classification system, as shown in Figure 1.

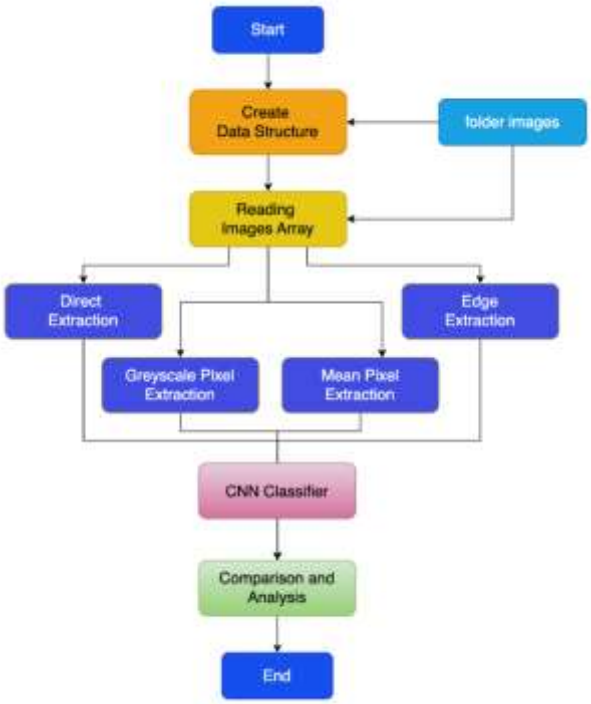


Figure 1. Research methodology framework

Figure 1 shows a few steps. It starts by creating a data structure from the images folder, processing the images, and loading them into an array. Next, based on four different image feature extraction techniques, the classification is based on a CNN classifier, and the results are conducted, compared, and analyzed.

Data Collection

The data used in this research is digital image data of fruit equipped with fruit-type labels obtained from the Kaggle webpage. Total number of images: 22,495. With a training set size of 16,854 images (one fruit or vegetable per image) and a test set size of 5,641 images (one fruit or vegetable per image). Number of classes: 33 (fruit and vegetables) with image size: 100x100 pixels. Training data file name format: [fruit/vegetable name]_[id].jpg (e.g. Apple Braeburn_100.jpg). Many images are also rotated to aid training. Test data file name format: [3 digit id].jpg (e.g., 001.jpg)

Feature Extraction

Feature extraction is a process for generating characteristics and reducing the dimensions of an object (in this case, an image object) from high to lower dimensions (Syufagi et al., 2011). This process will turn the most relevant features into a label. The most important thing in extracting features is analyzing the feature properties of an object and then organizing the numerical features into classes. In research conducted (Medjahed, 2015), several techniques for extracting features are mentioned, including:

Colour Features
The most commonly used technique is a color histogram, which looks at the distribution of colors in an image. Success in feature extraction does not depend on image size, image rotation, or enlarged images.

Texture Features
Texture is a collection of pixels that have certain characteristics. It refers to a visual pattern with homogeneous properties not produced from just one color. These features contain information about the structure that appears and its relationship with other pixels (Kunaver & Tasic, 2005). This technique has two categories: spatial features and spectral features.

Shape Features
Divided into two: feature extraction based on area and based on contour. Contour-based methods calculate features from the edges and ignore inner features, while area methods calculate all existing features.

Image Feature Selection Technique
Information from objects in digital images or image data can be used to identify or differentiate one image from another. The process of retrieving information from the image object is feature extraction, which produces parameters to define objects.

Grayscale Pixel Values
The simplest and most common way to create features from an image is to use grayscale or grey pixel values as separate features, as explained by (P.S. & Ramegowda, 2020). Consider the example shown in Figure 2.

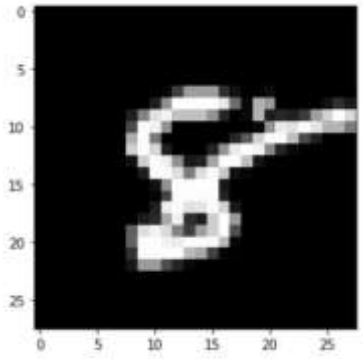


Figure 2. Example of a digital image of the number 8.

The number '8' shown has image dimensions of 28 x 28. The number of features will be the same as the number of times the pixel dimensions are multiplied, namely 784 features. To organize 784 pixels as features, we simply add each pixel value individually to produce a feature vector, as illustrated in Figure 2 below.

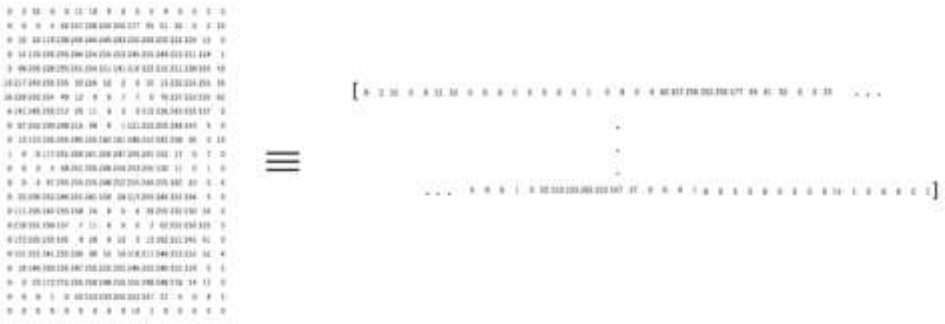


Figure 3. Illustration of the digital image of the number 8 and the features it forms.

Mean Pixel Value of Channels

Apart from reading digital images in grayscale mode, we can also read digital images in 3 color channel modes, namely red, green, and blue, or what is usually called RGB (Kumar & Bhatia, 2014).

As an illustration, we will get three matrices containing information on each channel's value, as shown in Figure 4 below.

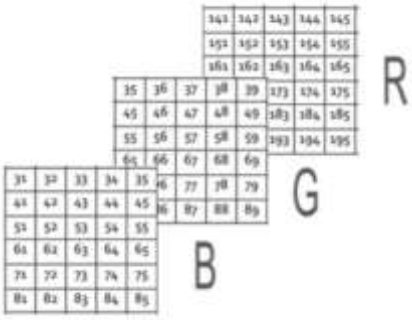


Figure 4. Three matrices for each RGB color band

Instead of using the pixel values of the three channels separately, we can generate a new matrix that has the average pixel values of the three channels, as shown in Figure 5

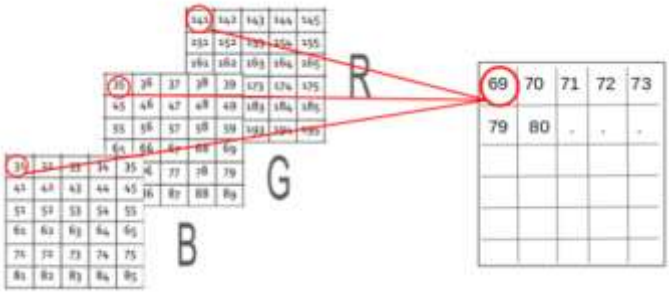


Figure 5. The new matrix is the average value of the three previous matrices

Furthermore, the features obtained can be used as input for the classification algorithm that will be used.

Extracting Edge Features

In the third method, we will use the color differences in the image as edges (Cui et al., 2008). As an illustration, suppose we have a digital image, as shown in Figure 6.

Figure 6. Color digital image

Using the principle of contrasting color differences found in one image, we will shape the edges and then get an image, as shown in Figure 7 below.

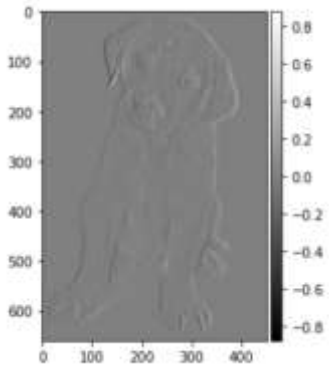


Figure 7. Image in edge format

Next, features are formed from the edge images obtained, which are then used as input material for the training and testing processes in the CNN algorithm. These three methods were chosen apart from being basic in the digital image feature extraction methods but also because these methods are easy to implement and understand the extraction process. By understanding the extraction work process from the basic method, it is hoped that it will be easier to understand and learn other more complicated and complex methods in the future.

Evaluation

Confusion Matrix is a table that analyses whether the classifier is good at recognizing tuples from different classes (Fibrianda & Bhawiyuga, 2018). If there are only two classes in the dataset, namely positive and negative, then several terms are used for measurement notation.

Results and Discussion

Based on the steps discussed in the research methodology, we first create a data structure and load the image into an array. Before constructing this data structure, it is essential to establish a clear folder structure of the image data. This involves showing the directory of category types of fruit images, which is crucial for ensuring that each category is correctly labeled and easily accessible for the subsequent preprocessing and model training stages.

As illustrated in Figure 8, the directory structure consists of several folders, each corresponding to a specific type of fruit. For instance, there are separate folders for apples, bananas, oranges, and other fruit categories. Within each folder, there are multiple images representing that particular fruit type. This hierarchical folder structure allows for efficient loading and management of the dataset, facilitating seamless integration into the image classification pipeline.

```
# Getting the Different class names from the directory

data_dir = pathlib.Path(train_path)
class_names = np.array(sorted([item.name for item in data_dir.glob('*')]))
print(class_names)
```

```
['Apple Braeburn' 'Apple Granny Smith' 'Apricot' 'Avocado' 'Banana'
 'Blueberry' 'Cactus fruit' 'Cantaloupe' 'Cherry' 'Clementine' 'Corn'
 'Cucumber Ripe' 'Grape Blue' 'Kiwi' 'Lemon' 'Limes' 'Mango' 'Onion White'
 'Orange' 'Papaya' 'Passion Fruit' 'Peach' 'Pear' 'Pepper Green'
 'Pepper Red' 'Pineapple' 'Plum' 'Pomegranate' 'Potato Red' 'Raspberry'
 'Strawberry' 'Tomato' 'Watermelon']
```

Figure 8. Getting the Different class names from the directory

The next step involves constructing a data structure to organize and manage the fruit image data efficiently. As illustrated in Figure 9, this data structure is designed to systematically categorize and store the images, ensuring that each image is associated with its corresponding fruit type. Once the data structure is established, the results of this organization process can be seen in Figure 10, which displays the hierarchical arrangement of the image data within the constructed data structure.

```
train_val_data = {'path' : [],
       'filename': [],
       'label': []}
for dirpath, dirnames, filenames in os.walk(train_path):
    for f in filenames:
        train_val_data['path'].append(dirpath)
        train_val_data['filename'].append(f)
```

Figure 9. Construct the data structure of the image dataset

	path	filename	label
0	/kaggle/input/fruit-recognition/train/train/Or...	Orange_398.jpg	Orange
1	/kaggle/input/fruit-recognition/train/train/Or...	Orange_20.jpg	Orange
2	/kaggle/input/fruit-recognition/train/train/Or...	Orange_337.jpg	Orange
3	/kaggle/input/fruit-recognition/train/train/Or...	Orange_190.jpg	Orange
4	/kaggle/input/fruit-recognition/train/train/Or...	Orange_456.jpg	Orange

Figure 10. The result of image data structure construction

Following the construction of the data structure, the images are loaded and converted into a format suitable for further processing. Specifically, each image is read from the data structure and transformed into a NumPy array. This conversion is essential for preparing the images for classification, as NumPy arrays are the standard input format for most machine-learning models. The process of loading and converting the images into NumPy arrays is demonstrated in Figure 11, where the array representation of an image is shown, ready for the classification processing stage.

```
images = []
label = []

for _, d in train_val_data_df.iterrows():
    img = load_img(os.path.join(d['path'],d['filename']))
    images.append(img_to_array(img))
    label.append(d['label'])
```

Figure 11. load and convert the image into a NumPy array

Balanced data is essential in machine learning because it directly influences the accuracy, fairness, and reliability of models. When datasets are balanced, models are trained with equal representation from all classes, preventing bias towards the majority class and ensuring that the model learns to recognize patterns from minority classes as well. This is particularly important in high-stakes applications, such as medical diagnostics and fraud detection, where incorrect predictions can have serious consequences.

Balanced data also improves the evaluation of models by providing realistic metrics like precision, recall, and F1 scores, which better reflect the model's true performance across all classes. It promotes ethical AI practices by ensuring that decisions are fair and do not disadvantage certain groups, thereby reducing discrimination. Overall, balanced data is crucial for developing effective and unbiased machine learning models that perform well in diverse, real-world scenarios. The frequency distribution of various fruit and vegetable labels in our dataset, are shown in Figure 12.

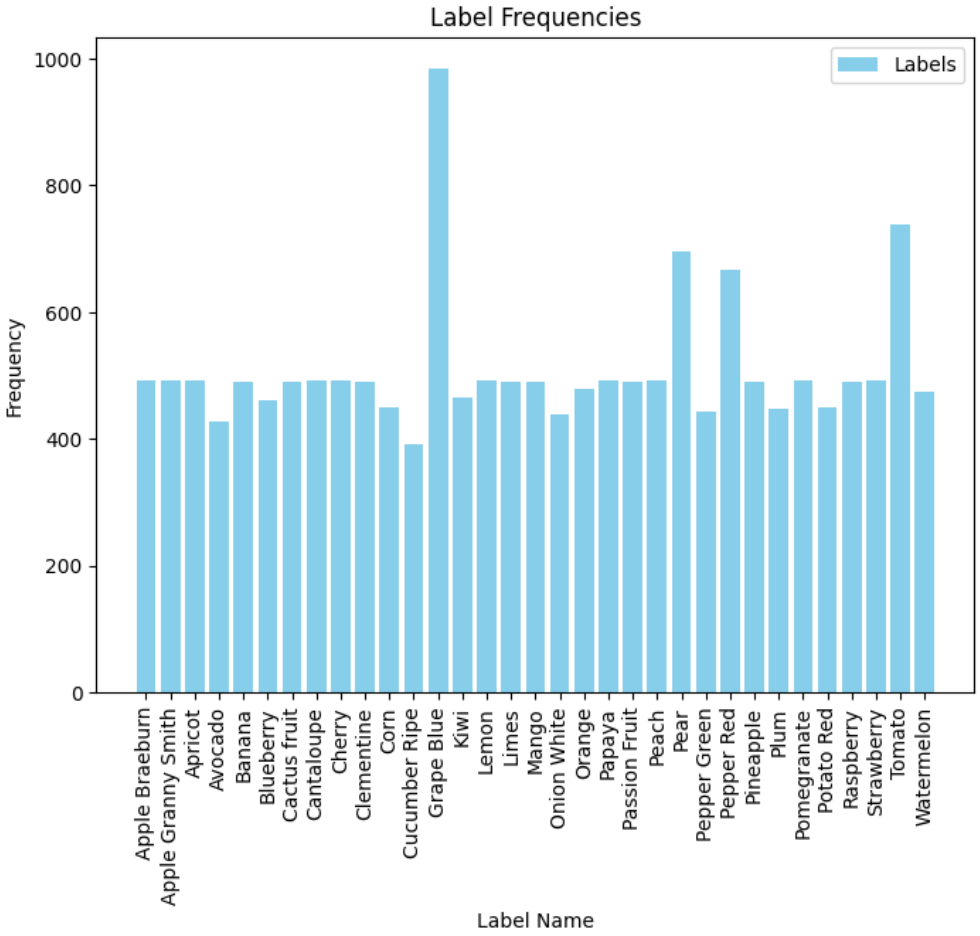


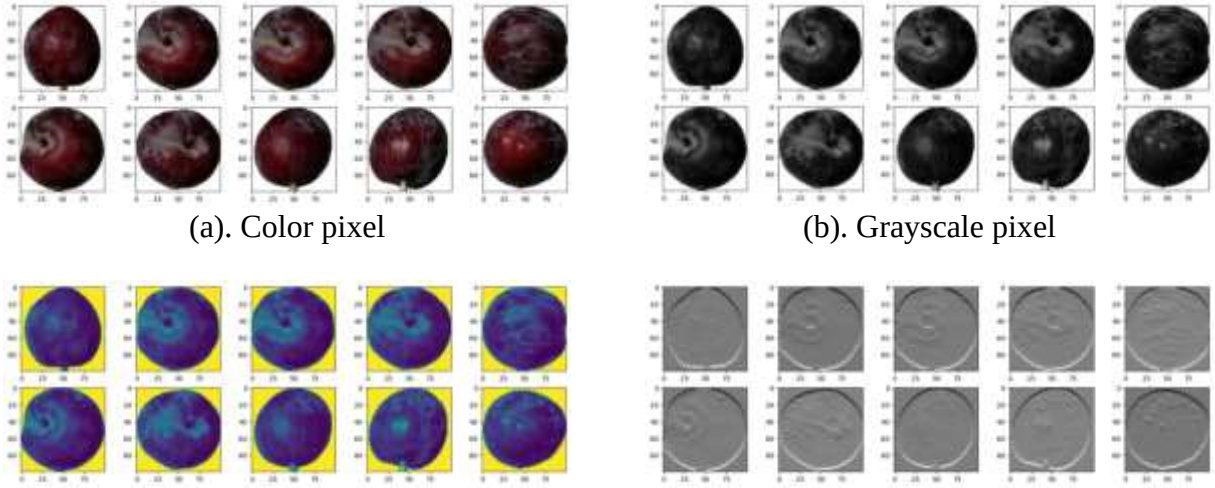
Figure 12. The frequency distribution of various fruit and vegetable labels.

The chart displays the frequency distribution of various fruit and vegetable labels, highlighting the imbalance in data representation among different categories. The x-axis lists the label names, including various fruits and vegetables like "Apple Braeburn," "Grape Blue," and "Tomato," while the y-axis shows the frequency of each label.

From Figure 12, it is evident that there is a noticeable variation in label frequencies, with "Grape Blue" having the highest count, approaching 1000, which indicates a significant overrepresentation compared to other labels. On the other hand, some labels like "Cactus Fruit" and "Onion White" have relatively lower counts, around 400-500, showing underrepresentation. This imbalance in label distribution can negatively affect machine learning model performance, as the model might become biased toward more frequent labels and struggle to accurately classify underrepresented ones.

The overall distribution highlights the need for balancing the data to ensure that the model can learn equally from all classes, thereby improving accuracy, fairness, and generalization in predictive tasks. Addressing this imbalance would involve either increasing the representation of underrepresented labels or employing techniques like data augmentation, oversampling, or undersampling to create a more balanced dataset.

Next, Figure 13 shows the sample images in color, grayscale, mean pixel, and edge pixel of image feature extraction.



(a). Color pixel (b). Grayscale pixel

(c). Mean pixel (d). Edge pixel

Figure 13. the sample images in four different image feature extraction

Figure 13 explains the image shows a set of visual representations of fruit under different preprocessing techniques commonly used in image analysis and machine learning tasks: (a) Color pixel, (b) Grayscale pixel, (c) Mean pixel, and (d) Edge pixel. Each representation highlights how image data can be manipulated to extract different features, impacting the way a model perceives and processes the input.

(a) Color Pixel: The first set of images showcases the fruit in its original color, preserving the full RGB information. This representation retains all the color details and textures, which are essential for tasks that rely on visual color cues, such as distinguishing different types of fruits. Color images provide rich data, allowing models to learn complex patterns based on color variations and textures, which are critical for accurate classification.

(b) Grayscale Pixel: The second set converts the color images into grayscale, reducing the information to a single-intensity channel. Grayscale images simplify the data by focusing solely on brightness variations rather than color, which can be beneficial in applications where color is less important, and texture or shape is the primary focus. This reduction in complexity can also speed up the processing time of models without significantly compromising accuracy, especially in scenarios where color is not a distinguishing feature.

(c) Mean Pixel: The third set, Mean Pixel representation, appears to involve applying a filter that smooths the image by averaging the pixel values in the surrounding areas. This technique reduces noise and highlights general patterns by softening edges and fine details. It is useful in preprocessing steps where the goal is to reduce image complexity and focus on broader structures, often used to emphasize larger features while ignoring smaller, less significant variations.

(d) Edge Pixel: The fourth set shows images with edge detection applied, which highlights the boundaries and contours of the fruit. This preprocessing method extracts the edges, enhancing the shapes and outlines of objects within the image. Edge detection is particularly useful in applications where shape and structure are critical, such as object recognition, segmentation, and boundary detection. By focusing on the edges, this technique emphasizes the geometrical features of the fruit, providing the model with clear outlines that can be key for distinguishing similar-looking objects.

Each preprocessing technique serves a unique purpose, enhancing specific aspects of the image data to improve the model's ability to learn relevant features. By selecting the appropriate representation, one can tailor the input data to better match the requirements of the machine learning task, ultimately enhancing the performance and accuracy of the model.

Once the datasets are prepared and organized, they must be split into training and testing sets to ensure the model's robustness and ability to generalize to unseen data. This step involves dividing the dataset such that 60% of the images are allocated for training, and the remaining 40% are reserved for testing, as illustrated in Figure 14. This split ensures that the model has sufficient data to learn from while retaining enough data to evaluate its performance objectively.

```
X_train, X_val, Y_train, Y_val =
    train_test_split(images, label_categorical, test_size = 0.4, random_state=101)

print("X train data : ", len(X_train))
print("X label data : ", len(X_val))
print("Y test data : ", len(Y_train))
print("Y label data : ", len(Y_val))
```

Figure 14. Split dataset 60-40.

Following the data split, a Convolutional Neural Network (CNN) model is created to perform the classification task. The CNN model is designed to learn from the training data and then validate its performance on the testing data. The process of creating and training the CNN model is depicted in Figure 15. This model architecture includes multiple layers, such as convolutional layers, pooling layers, and fully connected layers, which combine to extract features from the images and make accurate classifications. The model is trained using the training data, and its accuracy is validated using the testing data to ensure that it can generalize well to new, unseen images.

```
datagen = ImageDataGenerator(horizontal_flip=False,
                             vertical_flip=False,
                             rotation_range=0,
                             zoom_range=0.2,
                             width_shift_range=0,
                             height_shift_range=0,
                             shear_range=0,
                             fill_mode="nearest")

pretrained_model = tf.keras.applications.DenseNet121(input_shape=(100,100,3),
                                                      include_top=False,
                                                      weights='imagenet',
                                                      pooling='avg')
pretrained_model.trainable = False

def create_custom_cnn(input_shape, num_classes):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        MaxPooling2D(2, 2),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Flatten(),
        Dropout(0.5),
        Dense(512, activation='relu'),
        Dropout(0.2),
        Dense(num_classes, activation='softmax')
    ])
    return model

# Define the model parameters
input_shape = (100, 100, 3)  # Adjust based on your dataset preprocessing
num_classes = 33  # Adjust based on the number of fruit classes in your dataset

# Create the CNN model
model = create_custom_cnn(input_shape, num_classes)

optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['acc'])


# Assuming X_train and Y_train are already defined and preprocessed
history = model.fit(
    datagen.flow(X_train, Y_train, batch_size=64),
    validation_data=(X_val, Y_val),
    epochs=5
)
```

Figure 15. Model CNN creation processing

Figure 15 illustrates training and validating the Convolutional Neural Network (CNN) model. The provided Python code defines a custom Convolutional Neural Network (CNN) using the Keras Sequential API, designed for image classification tasks. The function create_custom_cnn() takes two parameters: input_shape, which specifies the dimensions of the input images, and num_classes, which denotes the number of output classes in the dataset. The model architecture begins with three convolutional layers with increasing filters (32, 64, and 128) to learn spatial hierarchies of features from the input images, each followed by a MaxPooling layer to reduce the spatial dimensions and down-sample the feature maps.

After the convolutional layers, the model uses a flattened layer to convert the 2D feature maps into a 1D vector, preparing the data for the dense layers. The dense section starts with a layer of 512 units using ReLU activation, which introduces non-linearity to help the model learn

complex patterns. Dropout layers are strategically included to prevent overfitting by randomly dropping neurons during training. The final layer is a dense layer with softmax activation, providing output probabilities for multi-class classification based on the number of classes specified.

The input shape is set to (100, 100, 3), indicating that the model is designed to handle 100x100 pixel RGB images, while num_classes is set to 33, matching the number of categories in the dataset. This CNN model is suitable for training on image classification tasks, with a robust architecture that incorporates dropout regularization to enhance generalization and mitigate overfitting, making it well-suited for real-world applications.

During the training phase, the CNN learns to recognize patterns and features in the fruit images by adjusting its weights and biases based on the training data. This process involves multiple epochs, where the model iteratively improves its predictions by minimizing the loss function through backpropagation and optimization algorithms such as Adam or SGD (Stochastic Gradient Descent).

The validation phase occurs simultaneously, where the model's performance is evaluated on the validation set after each epoch. This helps monitor the model's accuracy and detect any signs of overfitting, where the model might perform well on training data but poorly on unseen data. Figure 16 captures the essential steps and workflows involved in this training and validation process, highlighting the iterative nature of model development, as shown in Figure 16.

```
Epoch 1/5
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning:
  self._warn_if_super_not_called()
158/158 ━━━━━━━━━━━━━━━━━━━━ 74s 445ms/step - acc: 0.5063 - loss: 10.3170 - val_acc: 0.9899 - val_loss: 0.0296
Epoch 2/5
158/158 ━━━━━━━━━━━━━━━━━━━━ 71s 439ms/step - acc: 0.9556 - loss: 0.1411 - val_acc: 0.9740 - val_loss: 0.0822
Epoch 3/5
158/158 ━━━━━━━━━━━━━━━━━━━━ 70s 435ms/step - acc: 0.9536 - loss: 0.1673 - val_acc: 0.9975 - val_loss: 0.0119
Epoch 4/5
158/158 ━━━━━━━━━━━━━━━━━━━━ 70s 434ms/step - acc: 0.9738 - loss: 0.0885 - val_acc: 0.9932 - val_loss: 0.0194
Epoch 5/5
158/158 ━━━━━━━━━━━━━━━━━━━━ 71s 440ms/step - acc: 0.9690 - loss: 0.0964 - val_acc: 0.9941 - val_loss: 0.0162
```

Figure 16. Epoch training and validation processing

Initially, both accuracies might be low, but the training accuracy curve should rise as training progresses, indicating that the model is learning from the training data. The validation accuracy curve should increase if the model generalizes well to new data. The point at which these curves stabilize and converge signifies that the model has learned the underlying patterns in the data without overfitting.

Based on Figure 1, four distinct feature extraction techniques are implemented to enhance the performance of the fruit image classification task. These techniques are designed to preprocess the images and highlight important attributes that the classification algorithm can effectively utilize. The chosen techniques may include methods such as Grayscale Pixel Values, Mean Pixel Value of Channels, and Extracting Edge Features.

After extracting these features, the processed data are fed into Convolutional Neural Network (CNN) classifier algorithms. The CNN models leverage these features to effectively learn and classify the images into their respective fruit categories.

Table 1 summarizes the results of implementing these feature extraction techniques and deploying the data to CNN classifier algorithms. It provides a comparative analysis of the performance of each feature extraction technique when used in conjunction with the CNN classifiers, allowing for a clear evaluation of which feature extraction technique yields the best results regarding classification accuracy and overall model performance. Table 2, and. Table 3 show the convergence curve and confusion matrix, precision, recall, and F1-score, respectively.

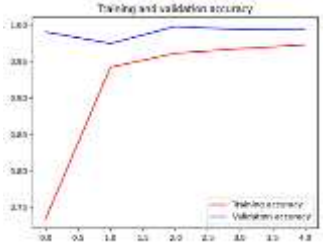
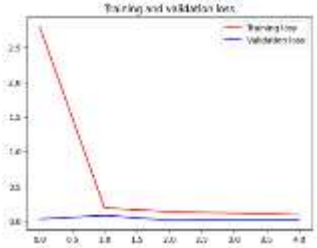
Table 1. The results and Comparison results.

Extraction Techniques	Accuracy (%)	Loss (%)	Val Accuracy (%)	Val Loss (%)
Colour Pixel	96.90	9.64	99.41	1.62
Grayscale Pixel	97.40	8.20	99.94	0.44
Mean Pixel	96.85	10.86	99.38	1.45
Edge Pixel	98.68	04.43	99.73	1.05

From comparison of different image feature extraction—Color Pixel, Grayscale Pixel, Mean Pixel, and Edge Pixel—reveals significant variations in model performance. Grayscale Pixel and Edge Pixel techniques demonstrate the best overall performance, with Grayscale achieving the highest validation accuracy (99.94%) and the lowest validation loss (0.44%), highlighting its effectiveness in simplifying data while retaining essential features. Edge Pixel also performs exceptionally well, emphasizing structural details like edges, which enhances both training and validation accuracy.

Color Pixel retains rich information but shows slightly higher loss due to the complexity of handling color data, which introduces noise and redundancy. Mean Pixel, which smooths the image by averaging values, reduces noise but might oversimplify important features, resulting in the highest training loss among the methods. Overall, Grayscale and Edge Pixel stand out as the most effective preprocessing techniques, balancing simplicity and feature retention to enhance model accuracy and generalization.

Table 2. The convergence curve of results.

Extraction Techniques	Training and Validation Accuracy (%)	Training and Validation Loss (%)
Colour Pixel		

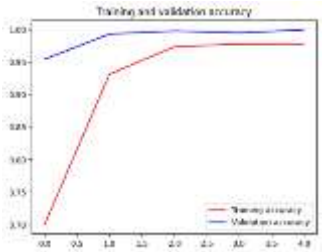
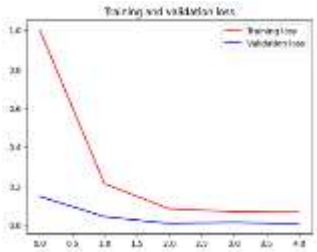
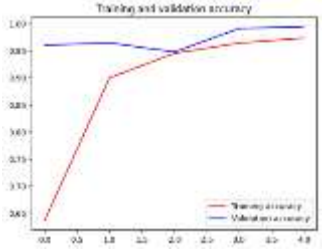
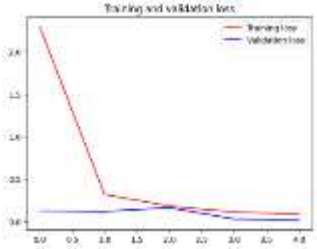
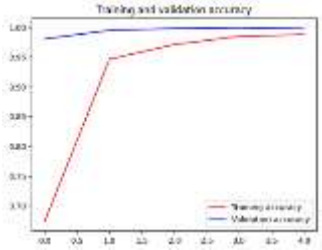
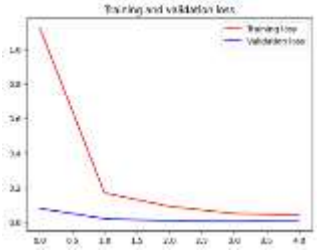
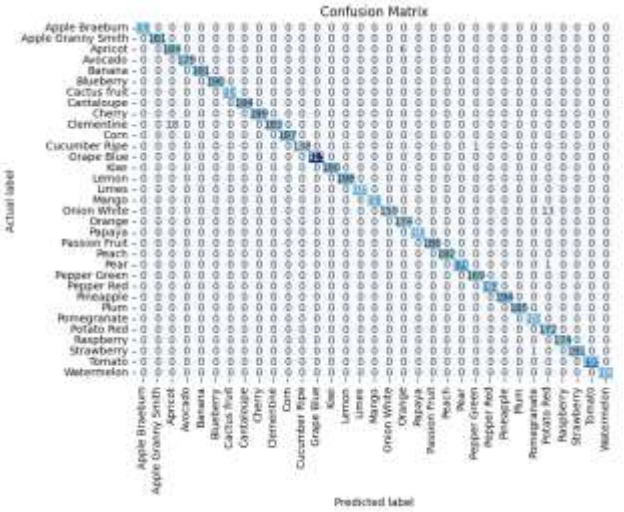
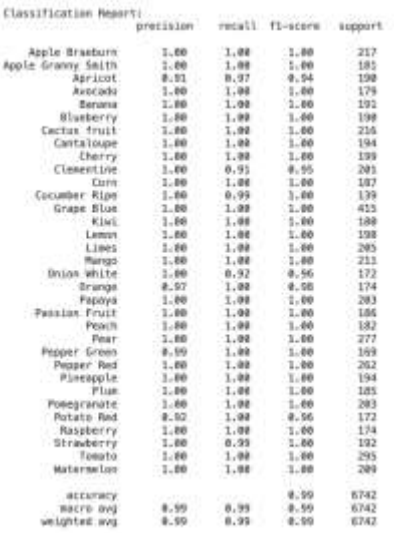
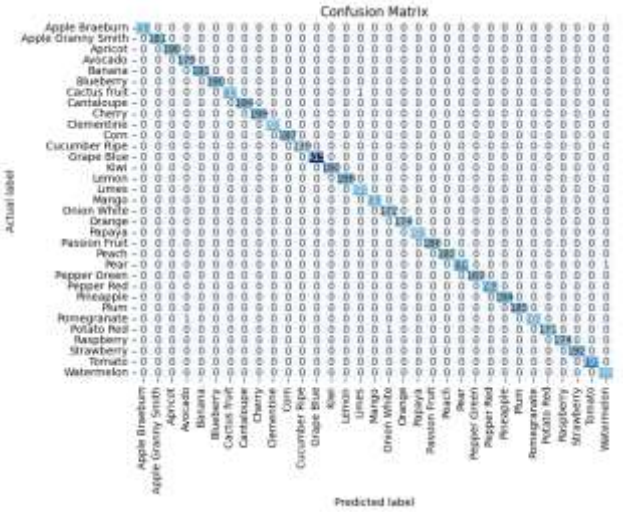
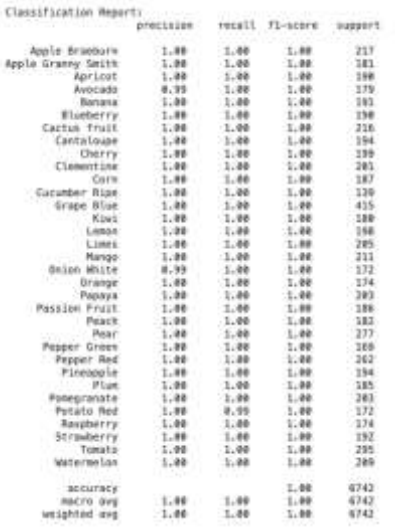
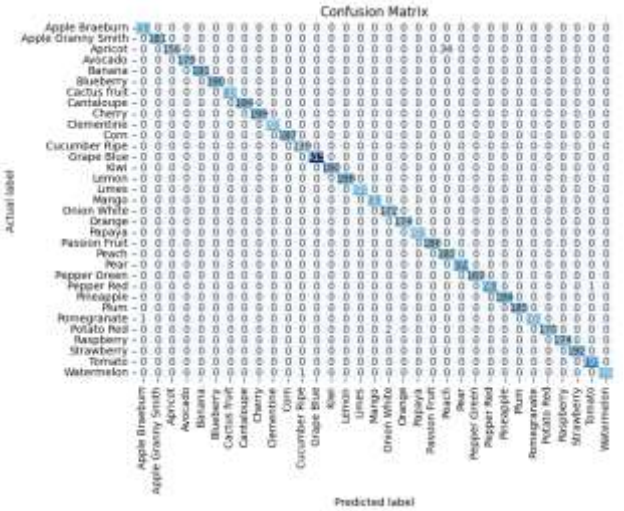
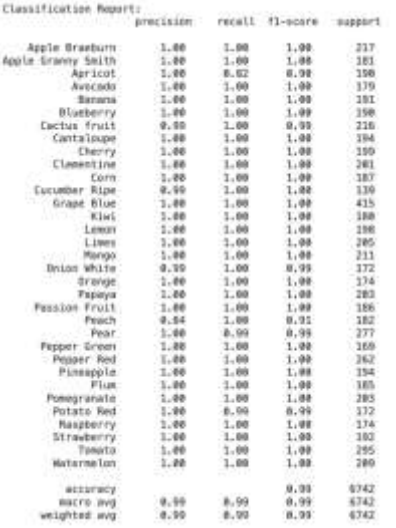
Grayscale Pixel		
Mean Pixel		
Edge Pixel		

Table 2 presents the analysis of training and validation accuracy and loss charts for different image feature extraction, which show distinct performance patterns. Grayscale Pixel and Edge Pixel techniques stand out for their excellent generalization, achieving high and closely aligned training and validation accuracy with low, stable loss values. Grayscale simplifies the input data, allowing the model to focus on essential features without being overwhelmed by color information, while Edge Pixel emphasizes structural contours, enhancing the model's ability to distinguish between different classes effectively.

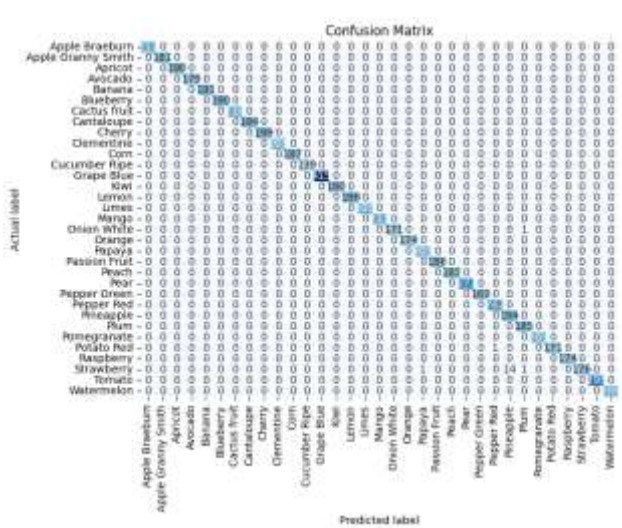
Color Pixel, although achieving high accuracy, shows signs of overfitting, as indicated by the volatility in validation loss, likely due to the complexity and noise introduced by color data. Mean Pixel also performs well but exhibits slightly higher validation loss, suggesting that its noise reduction approach might oversimplify the data, removing important details needed for precise classification.

Grayscale and Edge Pixel feature extraction methods provide the best balance between simplicity and feature retention, making them the most effective for training models that need to generalize well across unseen data. These techniques are particularly advantageous when model performance and generalization are critical.

Table 3. The confusion matrix and the precision, recall, and F1-score results.

Extraction Techniques		
Color Pixel		
Grayscale Pixel		
Mean Pixel		

Edge Pixel



Classification Report:

	precision	recall	f1-score	support
Apple Braeburn	1.00	1.00	1.00	217
Apple Granny Smith	1.00	1.00	1.00	181
Apricot	1.00	1.00	1.00	190
Avocado	1.00	1.00	1.00	179
Banana	1.00	1.00	1.00	191
Blueberry	1.00	1.00	1.00	198
Cactus Fruit	1.00	1.00	1.00	216
Cantaloupe	1.00	1.00	1.00	194
Cherry	1.00	1.00	1.00	199
Clementine	1.00	1.00	1.00	301
Corn	1.00	1.00	1.00	187
Cucumber Ripe	1.00	1.00	1.00	139
Grape Blue	1.00	1.00	1.00	415
Kiwi	1.00	1.00	1.00	190
Lemon	1.00	1.00	1.00	198
Limes	1.00	1.00	1.00	305
Mango	1.00	1.00	1.00	211
Onion White	1.00	0.99	1.00	172
Orange	1.00	1.00	1.00	174
Papaya	1.00	1.00	1.00	203
Passion Fruit	1.00	1.00	1.00	186
Peach	1.00	1.00	1.00	182
Pear	1.00	1.00	1.00	277
Pepper Green	1.00	1.00	1.00	169
Pepper Red	1.00	1.00	1.00	262
Pineapple	0.93	1.00	0.97	194
Plum	0.99	1.00	0.99	185
Pomegranate	1.00	1.00	1.00	283
Potato Red	1.00	0.99	1.00	172
Raspberry	1.00	1.00	1.00	174
Strawberry	1.00	0.92	0.96	192
Tomato	1.00	1.00	1.00	295
Watermelon	1.00	1.00	1.00	289
accuracy			1.00	6742
macro avg	1.00	1.00	1.00	6742
weighted avg	1.00	1.00	1.00	6742

Table 3 reveals how dataset imbalance affects model performance. The label frequency chart shows significant variations in class representation, with some classes like "Grape Blue" being highly overrepresented and others, such as "Cactus Fruit" and "Onion White," having fewer samples. This imbalance leads to models favoring more frequent classes, resulting in biased predictions.

Among the extraction techniques, Grayscale and Edge Pixel methods performed best, showing higher and more consistent classification scores. Grayscale extraction simplifies data to focus on essential features like texture and shape, while Edge Pixel extraction emphasizes structural contours, both of which help improve generalization. However, the presence of imbalanced data still affects underrepresented classes, causing lower precision, recall, and F1 scores for those categories.

Color Pixel extraction, while retaining full-color information, struggles more with imbalanced classes due to increased noise and complexity, leading to more misclassifications. Mean Pixel, which smooths the images, simplifies the input but sometimes at the cost of losing critical details, affecting performance for classes with fewer instances.

The analysis highlights that while certain extraction techniques can enhance model performance, addressing dataset imbalance through techniques like data augmentation or resampling is essential for achieving fair and robust classification across all classes.

By comparing these methods, an analysis can be carried out to determine which method provides the highest level of accuracy and the best overall performance for classifying fruit images using the CNN algorithm. This analysis helps in identifying the most effective feature extraction technique for developing a robust and efficient intelligent fruit selection system.

Conclusion

This research addresses the challenge of manual fruit selection in the agricultural sector by proposing a low-cost, computer vision-based intelligent fruit selection system. The study compared three feature extraction techniques—Colour Pixel Values, Grayscale Pixel Values, Mean Pixel Value of Channels, and Extracting Edge Features—to determine their effectiveness in classifying fruit images using a Convolutional Neural Network (CNN) algorithm.

The comparative analysis revealed that the Grayscale Pixel and Edge Pixel techniques demonstrate the best overall performance, with Grayscale achieving the highest validation accuracy (99.94%) and the lowest validation loss (0.44%), highlighting its effectiveness in simplifying data while retaining essential features. Edge Pixel also performs exceptionally well, emphasizing structural details like edges, which enhances both training and validation accuracy. These results underscore the importance of selecting appropriate feature extraction methods to enhance the accuracy and reliability of image classification systems.

The findings highlight the potential of edge detection as a powerful feature extraction technique for developing efficient and accurate intelligent fruit selection systems. By automating the fruit categorization process, this system can significantly reduce the time and labor required while ensuring consistent and precise classification. Future work could explore the integration of additional image preprocessing techniques and advanced CNN architectures to improve classification performance further.

This research contributes to the advancement of automated agricultural technologies, providing a viable solution to the challenges faced in manual fruit selection and paving the way for more sophisticated and scalable intelligent systems in the agricultural sector.

References

Cui, F., Zou, L., & Song, B. (2008). *Edge Feature Extraction Based on Digital Image Processing Techniques*. https://doi.org/10.1109/ICAL.2008.4636554

Fibrianda, M. F., & Bhawiyuga, A. (2018). Analisis Perbandingan Akurasi Deteksi Serangan pada Jaringan Komputer dengan Metode Naïve Bayes dan Support Vector Machine (SVM). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(9), 3112–3123.

Kumar, G., & Bhatia, P. (2014). *A Detailed Review of Feature Extraction in Image Processing Systems*. https://doi.org/10.1109/ACCT.2014.74

Kunaver, M., & Tasic, J. (2005). Image feature extraction - an overview. In *Journal of Crystal Growth - J CRYST GROWTH*. https://doi.org/10.1109/EURCON.2005.1629889

Madalina-Cosmina, P., & Sasu, L. (2014). Feature extraction, feature selection and machine learning for image classification: A case study. In *2014 International Conference on*

Optimization of Electrical and Electronic Equipment, OPTIM 2014. https://doi.org/10.1109/OPTIM.2014.6850925

Medjahed, S. A. (2015). A Comparative Study of Feature Extraction Methods in Images Classification. *International Journal of Image, Graphics and Signal Processing*, *7*, 16–23. https://doi.org/10.5815/ijigsp.2015.03.03

P, S., & Ramegowda, D. (2020). Gray Pixel Value based Vein Feature Extraction and Recognition on FPGA. *International Journal of Engineering Research And*, *V9*. https://doi.org/10.17577/IJERTV9IS050572

Ramprasath, M., Hariharan, S., & Prasath, R. (2022). *Image Classification using Convolutional Neural Networks*.

Rawat, W., & Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, *29*, 1–98. https://doi.org/10.1162/NECO_a_00990

Sharma, N., Jain, V., & Mishra, A. (2018). An Analysis Of Convolutional Neural Networks For Image Classification. *Procedia Computer Science*, *132*, 377–384. https://doi.org/https://doi.org/10.1016/j.procs.2018.05.198

Sharma, R., & Abrol, P. (2020). *Image feature extraction techniques*.

Soebroto, A. A. (2019). *Buku Ajar AI, Machine Learning & Deep Learning*. https://www.researchgate.net/publication/348003841

Syufagi, M. A., Hariadi, M., & Mauridhi Hery, P. (2011). A Cognitive Skill classification based on multi objective optimization using learning Vector Quantization for serious games. *ITB Journal of Information and Communication Technology*, *5*(3), 189–206. https://doi.org/10.5614/itbj.ict.2011.5.3.3

Xu, Z., Ahmad, S., Liao, Z., Xu, X., & Xiang, Z. (2023). Image feature extraction algorithm based on visual information. *Journal of Intelligent Systems*, *32*. https://doi.org/10.1515/jisys-2023-0111

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9