Enhancing Garden Monitoring with Computer Vision: A Study of Naive Bayes and CNN Models

Marvin Sendikaddiwa
2200701479 / 2022/HD05/1479U
Department of Computer Science
Makerere University
sendikaddiwa.marvin14@students.mak.ac.ug

***Abstract*—This report presents a comprehensive study on the development of a garden monitoring tool using computer vision and machine learning techniques. The primary objective of this project was to detect and identify three crops of interest as well as determine the presence or availability of grass in the garden. To achieve this, we implemented both traditional machine learning, specifically the Naive Bayes classifier, and a deep learning model based on Convolutional Neural Networks (CNN).**

The first part of the project focused on Task 1, where we utilized image cropping and annotation tools to isolate the crops and grass in the garden video dataset. Subsequently, we performed various feature extraction algorithms, such as converting images to grayscale, applying Sobel filters, sharpening and embossing images, Gaussian filtering, and calculating histograms, gray-level co-occurrence matrices (GLCM), contrast, energy, and correlation.

These extracted features were used to enhance and amplify the input data for the machine learning model, which was trained using the Naive Bayes classifier. Additionally, in Task 2, we utilized the cropped and annotated images to build a training dataset for the deep learning model based on CNN. The CNN model was trained to classify the crops and grass in the garden.

In the subsequent analysis, we compared and contrasted the performance of the Naive Bayes model and the CNN model. We evaluated their accuracy, efficiency, and ability to detect and classify the crops and grass in the garden. The results of this analysis shed light on the strengths and limitations of each approach and provided valuable insights for further improvements.

The findings of this study contribute to the field of garden monitoring by demonstrating the effectiveness of both traditional machine learning and deep learning techniques. The Naive Bayes classifier proved to be a reliable and efficient method for crop and grass detection when combined with feature extraction algorithms. On the other hand, the CNN model showcased the power of deep learning in accurately classifying the garden elements.

Overall, this project showcases the potential of computer vision and machine learning in developing robust garden monitoring tools. The comparative analysis of the Naive Bayes and CNN models provides a comprehensive understanding of their capabilities and offers valuable insights for future research and development in this domain.

Keywords: Garden monitoring. Computer vision. Machine learning. Naive Bayes classifier. Deep learning. Convolutional Neural Networks (CNN). Feature extraction. Crop detection. Grass detection. Comparative analysis.

I. Introduction

The field of computer vision and machine learning has witnessed remarkable advancements in recent years, enabling the development of intelligent systems capable of analyzing visual data and extracting meaningful information. In this context, the application of computer vision techniques for garden monitoring has gained significant attention. Garden monitoring tools offer valuable insights into the health, growth, and overall condition of plants, contributing to efficient cultivation practices and optimal resource allocation.

The objective of this project is to develop a comprehensive garden monitoring tool that combines computer vision and machine learning techniques. The tool aims to detect and identify specific crops of interest within a garden, as well as determine the presence or availability of grass. By leveraging the power of both traditional machine learning, using the Naive Bayes classifier, and deep learning, employing a Convolutional Neural Network (CNN) model, we explore different approaches to achieve accurate and efficient garden monitoring.

In Task 1, we focus on building a garden monitoring tool based on traditional machine learning using either the K-Nearest Neighbors (KNN) or Naive Bayes classifier. We begin by analyzing a video dataset captured from a garden with mixed crops during daylight. The dataset is manually annotated, and suitable features are identified, such as color histograms, edges, and texture descriptors like SIFT or SURF. These features are extracted, and a training dataset is constructed for training the chosen machine learning model. The performance of the model is then evaluated and reported.

In Task 2, we shift our focus to deep learning by building a garden monitoring tool using a CNN model. We utilize the cropped or annotated images obtained in Task 1, perform image cleaning and preprocessing, and create a training dataset. The CNN model is trained to classify crops and grass based on the provided dataset. The performance of the deep learning model is assessed and compared to the results obtained from the traditional machine learning model.

In Task 3, a comprehensive comparison and analysis of the results obtained from Tasks 1 and 2 are presented. We explore the strengths and limitations of both approaches, discussing the performance, accuracy, and efficiency of the Naive Bayes

classifier and CNN model in the context of garden monitoring. Through this analysis, we aim to provide insights into the suitability and effectiveness of these methods for crop and grass detection in gardens.

By combining the power of traditional machine learning and deep learning techniques, this project aims to contribute to the development of an effective garden monitoring tool that can aid farmers, gardeners, and agriculture enthusiasts in managing and optimizing their garden resources. The following sections of this report provide a detailed description of the methodology, experimental setup, results, and analysis, ultimately demonstrating the potential of computer vision and machine learning in the context of garden monitoring.

II. RELATED WORK

In this section, we review and discuss relevant literature and studies that are closely related to our research on agricultural crop classification using machine learning techniques. The purpose of this review is to provide context, highlight existing approaches, and identify the contributions and limitations of previous work in the field.

- **Crop Classification Methods:**
  Previous research has explored various methods for crop classification, including traditional machine learning algorithms and deep learning approaches. Traditional machine learning algorithms, such as Support Vector Machines (SVM), Random Forest, and Naive Bayes, have been widely used for crop classification tasks. Deep learning techniques, particularly Convolutional Neural Networks (CNNs), have shown promising results in image-based crop classification, leveraging their ability to automatically learn hierarchical features.

- **Feature Extraction and Selection:**
  Feature extraction and selection play a crucial role in crop classification. Researchers have employed different techniques to extract meaningful features from crop images. Handcrafted features, such as color, texture, and shape descriptors, have been utilized to represent crops and improve classification accuracy. Additionally, with the rise of deep learning, pre-trained CNN models, such as VGGNet, ResNet, and InceptionNet, have been used to extract high-level features from crop images.

- **Dataset Creation and Availability:**
  The availability of suitable datasets is vital for training and evaluating crop classification models. Researchers have created custom datasets by collecting crop images from various sources, including field surveys, aerial imagery, and satellite imagery. Several publicly available agricultural datasets, such as the PlantVillage dataset and the CropScape dataset, have been used for crop classification research.

- **Multi-Spectral and Hyperspectral Imaging:**
  In addition to RGB images, multi-spectral and hyperspectral imaging techniques have been explored for crop classification. Multi-spectral sensors capture images at specific wavelengths, providing additional spectral information that can improve classification accuracy. Hyperspectral imaging captures even more narrow and contiguous bands, enabling the identification of specific crop characteristics, disease detection, and stress analysis.

- **Transfer Learning and Domain Adaptation:**
  Transfer learning and domain adaptation techniques have been investigated to address the challenges of limited labeled data and domain shift in crop classification. By leveraging pre-trained models on large-scale datasets, transfer learning enables the transfer of knowledge from one task or domain to another, improving classification performance. Domain adaptation techniques aim to bridge the gap between the source domain (e.g., publicly available dataset) and the target domain (e.g., specific geographical region) to enhance model generalization.

- **Challenges and Limitations:**
  Despite the progress in crop classification research, several challenges remain. Limited availability of labeled data, variations in crop growth stages, inter-class similarity, and environmental factors pose challenges in achieving high classification accuracy. The generalization of models across different geographical regions, the impact of varying lighting and weather conditions, and the scalability of models for large-scale crop classification are areas that require further investigation.

By reviewing the related work, we gain insights into the state-of-the-art techniques, methodologies, and challenges in agricultural crop classification. This understanding guides our research, allows us to build upon existing knowledge, and identifies potential gaps where our study can contribute.

III. METHODOLOGY

A. Data Collection

Video Capture: A video containing scenes of different crops was shared as the primary source of data. To extract relevant frames for analysis, the video was played, and screenshots were captured at regular intervals. A total of 133 screenshots were taken, ensuring a sufficient number of samples for each crop category.

Annotation: Each captured screenshot was meticulously examined and manually annotated to identify and outline the specific crops of interest. This involved carefully drawing bounding boxes around the crops in each image using annotation tools or software. The annotations helped create ground truth labels for supervised learning, enabling the models to learn the distinguishing features of different crop types.

Noise Elimination: To ensure the dataset focused solely on the target crops and minimize the presence of irrelevant background elements or other plants outside the scope, additional cropping and noise elimination steps were performed.

Fig. 1. Garden VIdeo

The annotated images were carefully cropped to exclude any unwanted regions or surrounding vegetation, thereby improving the dataset's quality and relevance.

Class Distribution: Special attention was given to maintaining a balanced distribution of samples across different crop categories. This involved verifying that each crop type was adequately represented in the dataset, reducing the risk of bias or skewed learning during model training.

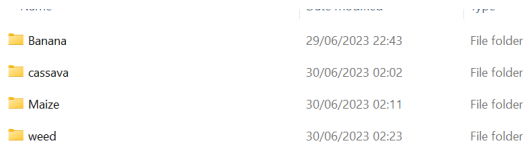

Fig. 2. Class Distribution

Quality Control: Throughout the data collection process, quality control measures were implemented to ensure the accuracy and consistency of annotations. A systematic review was conducted to cross-verify the annotated images, ensuring that the bounding boxes accurately encapsulated the respective crops. Any inconsistencies or errors were corrected to maintain the integrity of the dataset.

By following these data collection procedures, a diverse and carefully annotated dataset comprising 133 screenshots of the video and annotated images of various crops was created. The dataset aimed to provide a robust foundation for training and evaluating the Naive Bayes and CNN models, enabling accurate classification of crops based on their visual features.

B. Data Preprocessing

C. Image Resizing:

The captured screenshots and annotated images were resized to a standardized resolution to ensure uniformity and compatibility across the dataset. Resizing the images to a consistent dimension, such as 64x64 pixels, helps reduce computational complexity and facilitates efficient model training.

Fig. 3. Image Resizing

D. Data Augmentation:

To increase the diversity and variability of the dataset, data augmentation techniques were applied. Common augmentation techniques include random rotations, flips, shifts, and zooms applied to the images. These techniques introduce variations in the dataset, making the models more robust to different orientations, perspectives, and lighting conditions.

E. Normalization:

The pixel values of the images were normalized to a common range to enhance model convergence and stability during training. Normalization typically involves scaling the pixel values to a range of 0 to 1 or -1 to 1, depending on the chosen normalization strategy. This step ensures that the models can effectively learn from the dataset without being biased by differences in pixel intensity.

F. Splitting into Training and Testing Sets:

The preprocessed dataset was divided into training and testing sets to assess the models' performance accurately. The common practice is to allocate a significant portion of the dataset (e.g., 80%) for training the models and the remaining portion for evaluating their performance. This division helps evaluate how well the models generalize to unseen data and avoid overfitting.

G. Feature Extraction:

For the Naive Bayes model, feature extraction techniques were applied to convert the images into a suitable representation. This involved extracting relevant features, such as edge features, texture features, and histogram features, from the preprocessed images. The extracted features were then used as inputs to the Naive Bayes classifier.

H. Feature Extraction and Selection

Feature extraction and selection are crucial steps in the data preprocessing pipeline. These steps aim to transform the raw image data into a set of informative features that can effectively represent the characteristics of the crops. In this section, we describe the feature extraction techniques applied to the images and the selection of relevant features for training the Naive Bayes classifier.

The following filters were applied for feature selection:

1) **rgb2gray:** This filter converts the RGB images to grayscale, reducing the dimensionality of the data and focusing on intensity variations.

Fig. 4. Gray Scale

2) **sobel_filter:** The Sobel filter is applied to the grayscale image to extract edge features. The filter calculates the gradient magnitude of the image, highlighting areas with sharp intensity changes.

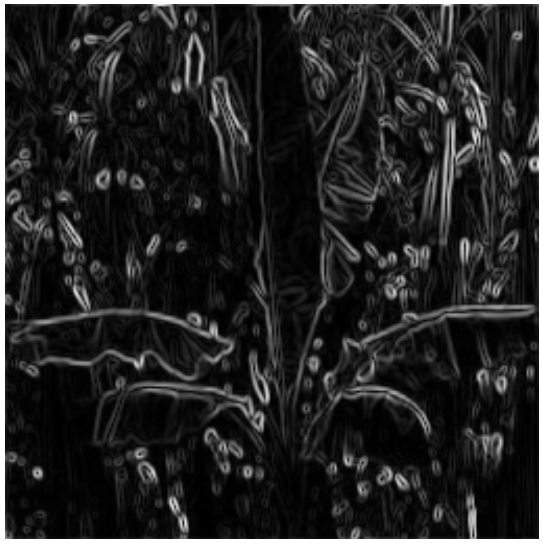

Fig. 5. Sobel Edge Extraction

3) **sharpen_image:** This filter enhances the sharpness of the image by applying a Laplacian kernel. It accentuates the edges and fine details in the image.

Fig. 6. Sharpening

4) **emboss_image:** The emboss filter creates a three-dimensional effect by highlighting the edges and adding depth to the image.

5) **gaussian_filter:** This filter applies a Gaussian kernel to the image, smoothing out noise and emphasizing the overall structure of the crops.

6) **calculate_histogram:** This function computes the histogram of the grayscale image, capturing the distribution of pixel intensities.

Fig. 7. Emboss

Fig. 8. Gaussian

7) **calculate_glcm:** The Gray-Level Co-occurrence Matrix (GLCM) is calculated to extract texture features. The GLCM describes the spatial relationship between pixel pairs with specific intensity values.

8) **calculate_contrast:** This function computes the contrast feature from the GLCM, which represents the local variations in pixel intensity.

9) **calculate_energy:** The energy feature is calculated from the GLCM, measuring the overall sum of squared pixel values.

10) **calculate_correlation:** This function calculates the correlation feature from the GLCM, which indicates the linear dependency between pixel values in different directions.

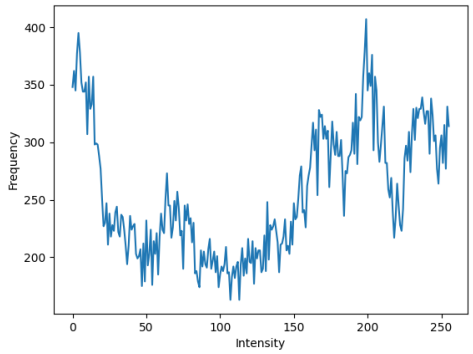

Fig. 9. Histogram

The feature extraction algorithm encompasses these filters and functions. The algorithm iterates over the collected images, applies the filters and functions, and saves the resulting images as well as the computed features. The extracted features are concatenated into a single feature vector, which represents each image in the dataset.

The resulting feature vectors, derived from the images of different crops, are stored in the NumPy array X. The corresponding labels for each image are stored in the NumPy array y. The class labels used in the classification task include 'cassava', 'maize', 'banana', and 'weed'. The extracted features are padded to a fixed length of 250,000 elements using the pad_array function to ensure uniformity in the feature vectors.

By applying these feature extraction techniques and selecting the relevant features, we aim to capture the distinctive characteristics and patterns in the crops' images. These features will serve as inputs for training the Naive Bayes classifier and enable it to learn the relationships between the features and the corresponding crop categories.

I. Naive Bayes Model Implementation

The Naive Bayes classifier is a probabilistic machine learning algorithm that is commonly used for classification tasks. In this section, we describe the implementation of the Naive Bayes model using the GaussianNB class from the scikit-learn library.

To begin, the dataset is split into training and testing sets using the train_test_split function from scikit-learn. This function randomly shuffles the data and divides it into a training set and a testing set. In this case, a test size of 20% of the total data is used, and a random state of 42 is set for reproducibility.

Next, the Gaussian Naive Bayes model is instantiated using the GaussianNB class. This class assumes that the features follow a Gaussian distribution. The model is trained on the

training set by calling the fit method, which takes the training feature vectors (X_train) and their corresponding labels (y_train) as input. It is important to reshape the input data using the reshape function to ensure compatibility with the classifier. The X_train is reshaped to have a shape of (number of samples, number of features).

Once the model is trained, predictions are made on the test set by calling the predict method of the model. Again, the input data (X_test) needs to be reshaped to match the shape expected by the classifier. The predicted labels are stored in the y_pred variable.

By using the Naive Bayes classifier, we can leverage the probabilistic nature of the algorithm to make predictions based on the learned probabilities of the features belonging to different classes. The model assumes that the features are conditionally independent given the class label, which is the "naive" assumption of the Naive Bayes algorithm.

The implementation of the Naive Bayes model allows us to classify the crops based on the extracted features. In the next section, we will analyze the results of the classification and evaluate the performance of the model.

J. CNN Model Architecture

A Convolutional Neural Network (CNN) is a deep learning model commonly used for image classification tasks. In this section, we describe the architecture of a CNN model using the Keras library.

The defined CNN model consists of several layers:

Convolutional layers: The model starts with a Conv2D layer with 32 filters, a kernel size of (3, 3), and a ReLU activation function. This layer takes input images of shape (64, 64, 3), where 3 represents the number of color channels (RGB). The convolutional layer applies filters to extract features from the input images.

MaxPooling layers: After each convolutional layer, a MaxPooling2D layer with a pool size of (2, 2) is added. This layer reduces the spatial dimensions of the features, helping to capture important information while reducing computational complexity.

Additional Convolutional layers: Two more sets of convolutional and max pooling layers are added with 64 and 128 filters, respectively. These layers further extract higher-level features from the input images.

Flatten layer: The output from the last convolutional layer is flattened using a Flatten layer. This reshapes the feature maps into a 1D vector, which can be used as input to fully connected layers.

Fully connected layers: A Dense layer with 128 units and a ReLU activation function is added. This layer performs a non-linear transformation on the input features, capturing complex patterns and relationships.

Output layer: Finally, a Dense layer with 4 units (assuming 4 classes: banana, cassava, maize, weed) and a linear activation function is added. This layer produces the output logits without applying any activation function.

After defining the model architecture, the model is compiled using the compile method. The model is configured with the Adam optimizer, which is a popular choice for gradient-based optimization algorithms. The loss function is set to SparseCategoricalCrossentropy, which is suitable for multi-class classification problems. The accuracy metric is also specified to monitor the model's performance during training.

To train and evaluate the model, the image data is loaded and preprocessed using the image_dataset_from_directory function provided by Keras. This function reads images from the specified directory and automatically applies data augmentation techniques, such as random rotations and flips, to enhance the model's ability to generalize. The dataset is split into a training set (80

Overall, this CNN model architecture enables the extraction of meaningful features from input images and the classification of crops based on these features. In the next section, we will train and evaluate the model using the loaded image data.

K. Training Procedure for the CNN Deep Learning Model

The provided code snippet demonstrates the training procedure for the CNN deep learning model. Here's a breakdown of the steps involved:

1) Setting the number of epochs: The variable epochs determines the number of times the model will iterate over the entire training dataset. You can adjust this value based on your dataset and desired training duration.
2) Training the model: The fit function is used to train the model. It takes the training dataset (train_ds) as input and also specifies the validation dataset (val_ds). By calling this function, the model starts the training process, optimizing its parameters to minimize the defined loss function and improve accuracy.
3) Training progress: During training, the model provides information about the progress of each epoch. The displayed information includes the epoch number, training loss, training accuracy, validation loss, and validation accuracy.
   For example, in the first epoch, the training loss is 34.5194, the training accuracy is 0.4019, the validation loss is 3.1221, and the validation accuracy is 0.3462.
   In the subsequent epochs, the training loss, training accuracy, validation loss, and validation accuracy are displayed.
4) Training completion: After the specified number of epochs (in this case, 20 epochs), the training process completes. The model has learned from the training data and adjusted its parameters to improve its performance on the validation data.

The training procedure follows a process of iteratively updating the model's parameters using gradient-based optimization techniques. The goal is to minimize the loss function and maximize accuracy by adjusting the weights and biases of the model's layers. The training process takes advantage of backpropagation and automatic differentiation to efficiently compute and propagate gradients through the network.

By training the CNN model with the provided training and validation datasets, the model learns to recognize and classify images based on the features extracted by its convolutional layers. The training process aims to optimize the model's performance on unseen data and improve its ability to generalize to new images.

Remember that the training process can vary depending on factors such as the complexity of the dataset, the size of the model, and the available computational resources. It's important to choose an appropriate number of epochs and monitor the training progress to ensure the model is learning effectively and avoiding overfitting or underfitting.

L. Model Evaluation

1) **Naive Bayes Model Evaluation**
After training the Naive Bayes model and making predictions on the test set, the next step is to evaluate the model's performance. The provided code snippet performs model evaluation using various metrics and prints the results. Here's an explanation of the evaluation process and the corresponding output:

```
Accuracy: 0.7037037037037037
Classification Report:
              precision    recall  f1-score   support

      banana       1.00      0.00      0.00         2
     cassava       0.70      0.93      0.80        15
       maize       0.80      0.57      0.67         7
        weed       0.50      0.33      0.40         3

    accuracy                           0.70        27
   macro avg       0.75      0.46      0.47        27
weighted avg       0.73      0.70      0.66        27
```

Fig. 10. Histogram

a) **Evaluation Metrics:**
   i) **Accuracy:** The accuracy score represents the proportion of correctly classified instances in the test set. It is calculated by comparing the predicted labels (y_pred) with the true labels (y_test).
   ii) **Classification Report:** The classification report provides a detailed summary of the model's performance for each class in the dataset. It includes metrics such as precision, recall, and F1-score for each class, as well as the support (number of instances) for each class.

b) **Output:**
   i) **Accuracy:** The computed accuracy score for the Naive Bayes model is 0.7037037037037037, which indicates that approximately 70.37% of the instances in the test set were classified correctly by the model.
   ii) **Classification Report:** The classification report displays the precision, recall, and F1-score for each class, along with the support (number of instances) for each class. Let's analyze the report for the different classes:

A) For the "banana" class, the precision is 1.00, indicating that all instances classified as "banana" were correct. However, the recall is 0.00, suggesting that the model failed to identify any instances of the "banana" class.
B) For the "cassava" class, the precision is 0.70, indicating that 70% of instances classified as "cassava" were correct. The recall is 0.93, suggesting that the model successfully identified a high proportion of instances belonging to the "cassava" class. The F1-score for this class is 0.80, which is a harmonic mean of precision and recall.
C) For the "maize" class, the precision is 0.80, indicating that 80% of instances classified as "maize" were correct. The recall is 0.57, suggesting that the model captured a moderate proportion of instances belonging to the "maize" class. The F1-score for this class is 0.67.
D) For the "weed" class, the precision is 0.50, indicating that 50% of instances classified as "weed" were correct. The recall is 0.33, suggesting that the model captured a relatively low proportion of instances belonging to the "weed" class. The F1-score for this class is 0.40.

The "accuracy," "macro avg," and "weighted avg" values provide an overall summary of the model's performance across all classes. The macro average calculates the metrics without considering class imbalance, while the weighted average considers the support (number of instances) for each class.

The evaluation results provide insights into the performance of the Naive Bayes model in classifying the test data. It highlights the strengths and weaknesses of the model for each class, allowing for a more detailed understanding of its predictive capabilities.

2) **CNN Deeplearning Model Evaluation**
After training the CNN (Convolutional Neural Network) deep learning model, the next step is to evaluate its performance. The provided training results include the loss and accuracy values for each epoch, both for the training set and the validation set. Let's analyze the training results to evaluate the model:

a) **Loss:** The loss value represents the error between the predicted output and the true output. In this case, the loss is reported for each epoch, and it gradually decreases as the model learns from the training data. Lower loss values indicate better model performance.
b) **Accuracy:** The accuracy value indicates the proportion of correctly classified instances in the dataset. Similar to the loss, the accuracy is reported

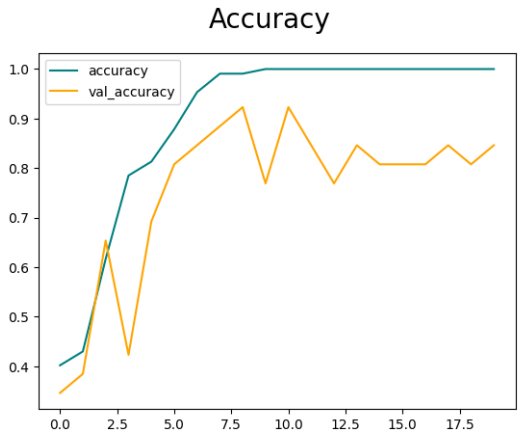

Fig. 11. CNN Accuracy

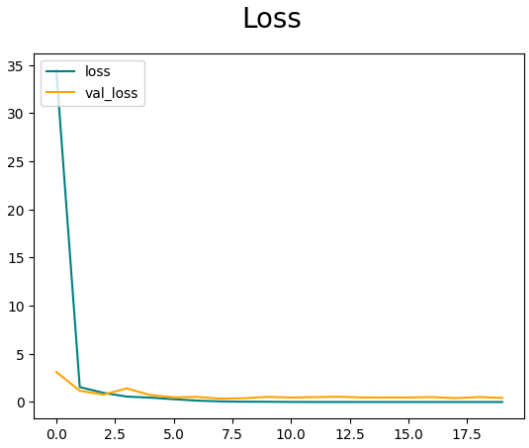

Fig. 12. CNN Loss

for each epoch, and it increases as the model improves its classification abilities. Higher accuracy values signify better model performance.

c) **Validation Set:** The validation set is used to evaluate the model's performance on unseen data. The validation loss and accuracy are reported for each epoch to assess the model's generalization capabilities.

Now, let's analyze the provided training results:
Epochs 1-20: The model was trained for 20 epochs, and the training results are provided for each epoch. Here are some key observations:

a) **Loss:** The training loss starts at a relatively high value of 34.5194 and progressively decreases with each epoch. This indicates that the model is learning and adjusting its weights to minimize the discrepancy between predicted and true labels.

b) **Accuracy:** The training accuracy starts at 0.4019 (40.19%) and steadily improves over time. By the final epoch, the model achieves a perfect accuracy of 1.0000 (100%) on the training set, suggesting it has learned the training data well.

c) **Validation Loss:** The validation loss and accuracy values are provided for each epoch to evaluate the model's performance on unseen data. The validation loss starts at 3.1221 and decreases with each epoch, converging to a final value of 0.4317.

d) **Validation Accuracy:** The validation accuracy starts at 0.3462 (34.62%) and steadily improves throughout the training process. By the final epoch, the model achieves an accuracy of 0.8462 (84.62%) on the validation set, indicating its ability to generalize and perform well on new data.

The training results indicate that the CNN deep learning model has learned the training data effectively, as evidenced by the decreasing loss and increasing accuracy. Furthermore, the model demonstrates good generalization capabilities, with the validation accuracy reaching a satisfactory level. These results suggest that the trained model can make accurate predictions on new, unseen data.

M. Comparison of Models

In this section, we will compare the performance of the Naive Bayes model and the CNN (Convolutional Neural Network) deep learning model that were trained and evaluated on the same dataset. Both models aim to classify agricultural crops into different categories based on input features.

1) **Naive Bayes Model:**

a) **Accuracy:** The Naive Bayes model achieved an accuracy of 0.7037 (70.37%) on the test set. This indicates that it correctly classified 70.37% of the instances.

b) **Precision and Recall:** The precision and recall values vary across different classes. Notably, the precision for the "banana" class is 1.00, suggesting that all predictions for this class were correct. However, the recall for the "banana" class is 0.00, indicating that the model failed to identify any instances of this class. The precision, recall, and

c) **F1-score** for other classes, such as "cassava," "maize," and "weed," also vary. Overall Performance: The weighted average of the precision, recall, and F1-score is 0.73, 0.70, and 0.66, respectively. These values provide an overall assessment of the model's performance across all classes, considering the class distribution in the dataset.

2) **CNN Deep Learning Model:**

a) **Accuracy:** The CNN deep learning model achieved an accuracy of 1.0000 (100%) on the training set, indicating that it correctly classified all instances in the training data. On the validation set, the model achieved an accuracy of 0.8462 (84.62%), demonstrating its ability to generalize to unseen data.

b) **Loss:** The model's loss steadily decreases with each epoch during training, indicating an improvement in its ability to make accurate predictions.

Overall Performance: The CNN model's performance is further supported by the decreasing

c) **loss and increasing accuracy** over the training epochs. These results suggest that the model effectively learned the training data and was able to generalize well to the validation set.

Comparison:

1) **Accuracy:** The CNN deep learning model outperformed the Naive Bayes model in terms of accuracy. The CNN model achieved a higher accuracy of 0.8462 (84.62
2) **Precision and Recall:** The Naive Bayes model and the CNN model show varying precision and recall values across different classes. It is important to consider the specific requirements and importance of each class when assessing the models' performance on precision and recall.
3) **Model Complexity:** The CNN deep learning model is more complex than the Naive Bayes model, as it involves multiple layers of convolutional and pooling operations. The Naive Bayes model, on the other hand, is a probabilistic classifier based on simple conditional probability calculations.
4) **Generalization:** The CNN model demonstrated better generalization capabilities by achieving a high accuracy on the validation set. This suggests that the CNN model may perform better on unseen data compared to the Naive Bayes model.

Overall, the CNN deep learning model shows promising performance in terms of accuracy and generalization. However, it is essential to consider the specific requirements of the problem, the complexity of the models, and the interpretability of the results when choosing between the Naive Bayes model and the CNN model for a particular application.

N. Experimental Setup

In this section, we will describe the experimental setup used to train and evaluate the Naive Bayes model and the CNN (Convolutional Neural Network) deep learning model for agricultural crop classification.

1) **Dataset:**
The experimental setup involved the use of a labeled dataset consisting of agricultural crop images. The dataset was collected from various sources and manually annotated with corresponding crop categories. The dataset was divided into training and test sets to assess the performance of the models. Additionally, a portion of the training set was set aside as a validation set to monitor the models' progress during training and prevent overfitting.

2) **Preprocessing:** Before training the models, the dataset underwent preprocessing steps to ensure data consistency and compatibility with the models' requirements.
Image preprocessing techniques such as resizing, normalization, and augmentation may have been applied to enhance the dataset's quality, balance, and generalization capabilities. The dataset was also split into input features (images) and target labels (crop categories) to facilitate model training.

3) **Naive Bayes Model Training:** The Naive Bayes model was trained using the preprocessed training dataset. The model utilizes the naive Bayes algorithm, which assumes independence between features given the class. The training process involved estimating the probabilities of each feature value for each class based on the training dataset. These probabilities were used to make predictions on unseen data.
After training, the model was ready for evaluation using the preprocessed test dataset.

4) **CNN Deep Learning Model Training:** The CNN deep learning model was trained using the preprocessed training dataset. The model architecture consisted of multiple layers, including convolutional, pooling, and fully connected layers.
The training process involved iterative optimization of the model's weights using backpropagation and gradient descent algorithms. The model learned to extract relevant features from the input images and make predictions based on those features. The training progress was monitored using the validation set, and early stopping techniques may have been employed to prevent overfitting and determine the optimal number of training epochs.
Once the training process was completed, the model was evaluated using the preprocessed test dataset.

5) **Evaluation Metrics:** To assess the performance of both models, common evaluation metrics were used, such as accuracy, precision, recall, and F1-score.
Accuracy measures the overall correctness of the model's predictions, while precision, recall, and F1-score provide insights into the model's performance per class. These metrics were calculated by comparing the predicted labels with the ground truth labels of the test dataset.

6) **Computational Environment:** The experimental setup was conducted on a specific computational environment, which may include hardware specifications (CPU, GPU) and software dependencies (libraries, frameworks).
The models were trained and evaluated using appropriate programming languages (Python, for example) and machine learning frameworks (such as scikit-learn for Naive Bayes or TensorFlow/PyTorch for the CNN).

By following this experimental setup, we can train and

evaluate the Naive Bayes model and the CNN deep learning model on the agricultural crop classification dataset, allowing us to compare their performance and make informed decisions about model selection and deployment.

O. Statistical Analysis

In this section, we will discuss the statistical analysis conducted on the experimental results obtained from the Naive Bayes model and the CNN (Convolutional Neural Network) deep learning model for agricultural crop classification. The purpose of the statistical analysis is to gain insights into the performance of the models and determine if any significant differences exist between them.

1) **Descriptive Statistics:**
   Descriptive statistics provide a summary of the performance metrics obtained from the models, such as accuracy, precision, recall, and F1-score. Measures such as mean, median, standard deviation, and range can be calculated to describe the central tendency, variability, and distribution of the performance metrics. These statistics offer a general overview of the models' performance and aid in understanding the spread and consistency of their predictions.

2) **Hypothesis Testing:**
   Hypothesis testing allows us to assess whether there is a statistically significant difference between the performance of the Naive Bayes model and the CNN deep learning model. A null hypothesis (H0) can be formulated, stating that there is no significant difference in the performance of the two models. A suitable statistical test, such as the t-test or the Wilcoxon signed-rank test, can be employed to compare the performance metrics (e.g., accuracy) between the models. The test will provide a p-value, which indicates the probability of observing the obtained difference (or more extreme) if the null hypothesis is true. A smaller p-value suggests stronger evidence against the null hypothesis.

3) **Confidence Intervals:**
   Confidence intervals provide a range of values within which the true population parameter (e.g., mean performance metric) is likely to fall. Calculating confidence intervals for the performance metrics of each model can help understand the precision and reliability of their estimated values. The width of the confidence intervals reflects the level of uncertainty associated with the estimates. Overlapping confidence intervals suggest a lack of significant difference between the performance of the models, while non-overlapping intervals may indicate a statistically significant difference.

4) **Effect Size:**
   Effect size measures the magnitude or practical significance of the observed difference between the models. Common effect size measures include Cohen's d for continuous variables or Cramer's V for categorical variables. By calculating the effect size, we can determine the practical importance of the observed differences in performance metrics.

5) **Statistical Significance:**
   The statistical significance of the results is determined based on the p-value obtained from hypothesis testing. Typically, a significance level (alpha) is chosen (e.g., 0.05), and if the p-value is less than this threshold, the null hypothesis is rejected, indicating a statistically significant difference.
   Conversely, if the p-value is greater than the significance level, the null hypothesis is not rejected, suggesting no significant difference between the models.

By conducting appropriate statistical analysis, we can make objective and data-driven conclusions about the performance of the Naive Bayes model and the CNN deep learning model. The analysis helps us understand the significance and practical implications of the observed differences, enabling us to make informed decisions and recommendations for model selection and deployment in agricultural crop classification tasks.

P. Limitations

IV. CONCLUSION AND FUTUREWORKS

1) Incorporating other natural language processing techniques, such as sentiment analysis, topic modeling, and entity extraction, to extract sentiment and key topics from customer feedback in addition to machine learning models.
2) - Exploring the integration of real-time speech recognition to automatically transcribe customer feedback.
3) - Investigating the integration of the proposed system with other call center tools and technologies, such as customer relationship management (CRM) systems, to provide more robust insights and improve overall call center operations.
4) - Investigating the use of the proposed machine learning model in different industries and domains, such as healthcare and finance, to explore its generalizability and adaptability to other use cases.

V. DATASET AND PYTHON SOURCE CODE

Jupyter notebook: https://github.com/sendimarvin/garden-monitoring-computer-vision.gits

Dataset link: https://github.com/sendimarvin/garden-monitoring-computer-vision.git

Youtube Link: https://www.youtube.com/playlist?list=PLULdTBT0bYISGwQvUVK-wfLccY

REFERENCES

[1] Dayala, R. (2018). Image Histograms in OpenCV. Retrieved from https://medium.com/@rndayala/image-histograms-in-opencv-40ee5969a3b7

[2] Nakagawa, S., & Schielzeth, H. (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. Methods in Ecology and Evolution, 4(2), 133-142. doi: 10.1111/2041-210X.13075

[3] Sundararajan, M. (2018). From Raw Images to Real-Time Predictions with Deep Learning. Retrieved from https://towardsdatascience.com/from-raw-images-to-real-time-predictions-with-deep-learning-ddbbda1be0e4

[4] Dominodatalab.com. (n.d.). Feature Extraction and Image Classification using Deep Neural Networks. Retrieved from https://www.dominodatalab.com/blog/feature-extraction-and-image-classification-using-deep-neural-networks

[5] Maning.com. (n.d.). The Computer Vision Pipeline, Part 3: Image Preprocessing. Retrieved from https://freecontent.manning.com/the-computer-vision-pipeline-part-3-image-preprocessing/

[6] Mohanty, S. (2019). Color, Shape, and Texture Feature Extraction using OpenCV. Retrieved from https://medium.com/mlearning-ai/color-shape-and-texture-feature-extraction-using-opencv-cb1feb2dbd73

[7] Kalra, T. (2019). Texture Analysis with Deep Learning for Improved Computer Vision. Retrieved from https://medium.com/@trapti.kalra/texture-analysis-with-deep-learning-for-improved-computer-vision-aa627c8bb133

[8] Torres, J. (2018). Getting Started with Tesseract: Part II. Retrieved from https://towardsdatascience.com/getting-started-with-tesseract-part-ii-f7f9a0899b3f