Image-based Fruit Classification

Team: Joker

Team Members: Jiacheng Zhou, Ruitao Shen, Shihang Liu,
Siwei Guo, Yuanchi Guo

Content:

- Motivation
- Goals
- Literature review
- Data description and exploratory analysis
- Preliminary research ideas and future plan
- Implication
- Acknowledgement
- Citation

Motivation:

In the fast-paced world of grocery retail, accurately identifying produce is crucial for both customers and sellers. For sellers, precise and up-to-date monitoring of sales and inventory of various products is essential for balancing revenue and losses, particularly for perishable items like fruit. Yet, certain fruits bear such striking resemblances that distinguishing between them becomes challenging, leading to labeling errors and confusion for all parties involved. For instance, fruits like honey crisp and gala apples, or parsley and cilantro, can easily be mistaken for one another, resulting in pricing discrepancies and undermining trust between buyers and sellers.

To tackle these challenges, we propose a research project focused on using machine learning to identify fruits from images. With machine learning, our aim is to develop a smart system capable of distinguishing between different types of fruits simply by analyzing their images. Such a system will not only streamline tasks for grocery stores but also instill confidence in customers. Market managers can also utilize such a system to regulate intentional or unintentional mislabeling of products.

In the upcoming sections of this research proposal, we delve into the objectives, methodologies, and models, as well as the anticipated outcomes of our fruit image identification project. Through the utilization of diverse machine learning models and validation techniques, along with statistical analysis to assess accuracy and confidence levels in identification, our goal is to directly offer practical solutions for enhancing the accuracy and efficiency of fruit identification. Additionally, it can serve as a valuable case study for broader applications of multiple product identification across the industry.

Goals:

We want to design and implement a machine learning model to classify various fruits efficiently and accurately by analyzing their visual features in images. For example, Shishito and Padrón peppers can sometimes look very similar, with both peppers having a green and wrinkled appearance, and Nectarines and Peaches can look very similar, even though they have completely different textures which can lead to misclassification errors when managed by machines. Therefore, machines will cause inefficiency and misidentifying when customers want to check out. Moreover, if customers want to use their phones to scan certain types of fruits or vegetables for nutrition, misinformation may cause dietary problems. The project aims to utilize advanced image processing and deep learning techniques to ensure robust performance even under different lighting, orientation, and scale conditions. The goal is to achieve high accuracy and reliability in distinguishing between different fruit categories. This will facilitate applications such as self-checkout in supermarkets, agricultural categorization, dietary tracking, etc.

Literature Review:

In this part, we found three papers from Google Scholar, which have been contributed in various ways. One paper directly discusses fruit classifications and another about fruit grade classifications. These two directly gave us an intuition of how we should conduct this research since our projects are highly correlated. At the same time, the other one acts as insight and provides approaches through deep learning to classify wildlife. This one may not be directly correlated, but it provided us valuable insights into the application of deep learning for image classification tasks, which can be extrapolated in fruit classification as well.

The following are the main points that provided us with instructive ideas:
Feature extraction and representation: One of these papers highlighted the

transformative impact of deep learning in automating fruit classification and grading. The shift from manual inspection to AI systems has provided significant improvements in accuracy, efficiency, and scalability. We can see the importance of accurately extracting and representing features such as color, texture, size, and shape, which are critical for distinguishing fruit categories. Here is an example of that:

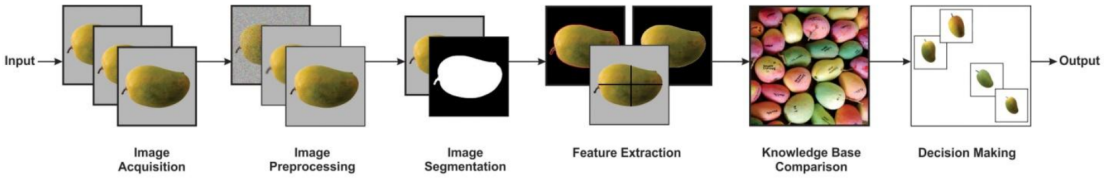

Figure 1: example of accurately attracting

Deep learning architectures: The effectiveness of CNNs and other deep learning models, for example, in learning complex hierarchical feature representations from fruit images, leads to superior classification performance.

Model Interpretability: Techniques such as Grad-CAM provide insight into the decision-making process of CNNs, helping to fine-tune the models and understand their limitations.

This is the process from the animal detection paper:

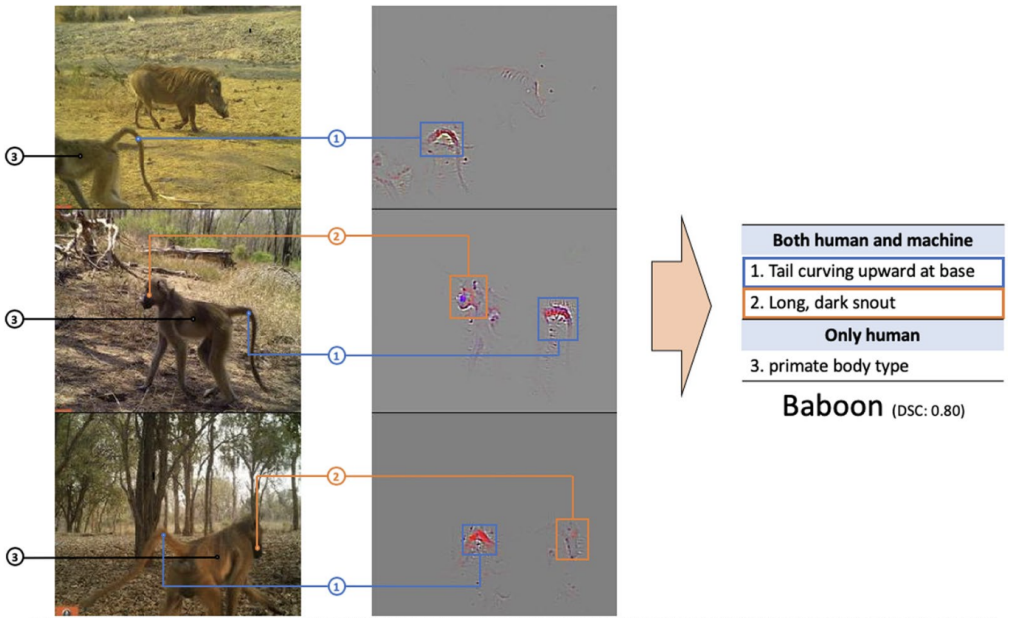

Figure 2: process of animal detection

GG-CAM generated local discriminative visual features of randomly selected Baboon images. In classifying the Baboon, CNN focused on the face and tail. Most of the features extracted by the CNN have a counterpart (the similar focal visual component) in the human visual descriptors (indicated by color and agreed by at least 2 of the 4 authors). The similarity was calculated as the DSC between the extracted features and the corresponding human descriptors.

Practical applications of these models in industrial settings highlight the potential for significant return on investment through automation, consistent quality assessment, and waste reduction. The integration of deep learning models in fruit classification systems represents a significant advancement in agricultural automation and promises to enhance quality control, optimize supply chains, and reduce labor dependency. This will benefit humans more by saving them from meaningless and tedious processes.

Specifically, we see that numerous types of fruit are similar in shape, color, and texture. There is also a great deal of variation between fruits of the same group, depending on the stage of ripeness of the fruit as well as the form in which the fruit is presented. Future research directions may focus on improving the robustness of these systems under different environmental conditions, incorporating multispectral imaging, and addressing challenges related to fruit diversity and anomalies.

Data description and exploratory analysis:

Our original fruit image dataset was sourced from Kaggle, https://www.kaggle.com/datasets/moltean/fruits, created by Mihai Oltean and last updated two years ago [1]. The dataset comprises 90,483 images, each depicting a single fruit or vegetable. It encompasses a total of 131 classes representing various fruits and vegetables. We selected this dataset due to its unique characteristic of having a noise-free background in all images. During the photography process, the

author deliberately used a white sheet of paper as the background and employed a dedicated algorithm to separate the fruit from the background, mitigating the impact of lighting variations on image quality.

The original dataset was divided into test, train, and validation folders for the author's research purposes. We merged these folders into one and subsequently divided them into our own train, test, and validation datasets (Figure 3).

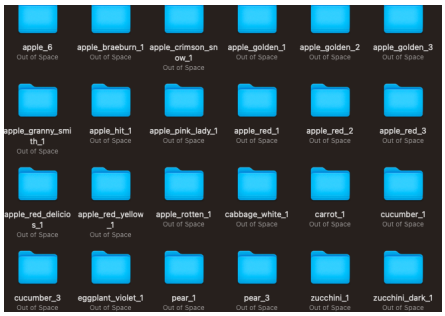

Figure 3: Image Folder Review

We employed RStudio as our primary tool throughout the project. The image loading and preparation process involved utilizing the 'jpeg' and 'grid' libraries to display a collection of images stored in a specified directory. The procedure began with setting the path to the image directory and retrieving the list of image files. For each image, the script read the JPEG file, set up an empty plot, and rasterized the image onto the plot. It accommodated cases where there were fewer than 15 images in the folder by displaying as many images as available and notifying the user accordingly. Finally, the script reset the plot layout to its default configuration, as depicted in Figure 4.

```r
```{r, fig_height=10, fif_width = 5}
library(jpeg)
library(grid)

image_dir <- "/Users/guosiwei/Desktop/All images/"
image_folders <- list.dirs(image_dir, full.names = FALSE, recursive = FALSE)
folder <- image_folders[1]
image_files <- list.files(file.path(image_dir, folder), full.names = TRUE)

par(mfrow=c(3, 5), mar=c(0, 0, 0, 0))
for (i in 1:15) {
    if (i <= length(image_files)) {
        img <- readJPEG(image_files[i])
        plot(1, type='n', xlab='', ylab='', xlim=c(0, 1), ylim=c(0, 1))
        rasterImage(img, 0, 0, 1, 1)
    } else {
        cat("Not enough images in the folder.\n")
        break
    }
}

par(mfrow=c(1, 1))
```
```

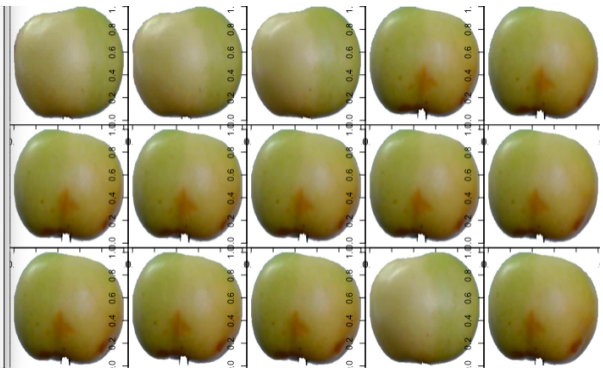

Figure 4: R Code to Display Images and 15 Examples of Images

To prepare data for image classification analysis, we used Principal Component Analysis (PCA) to reduce dimension. We utilized the ‘pracma’ library. We first applied PCA on seven images to see the reconstructed images (Figure 5).

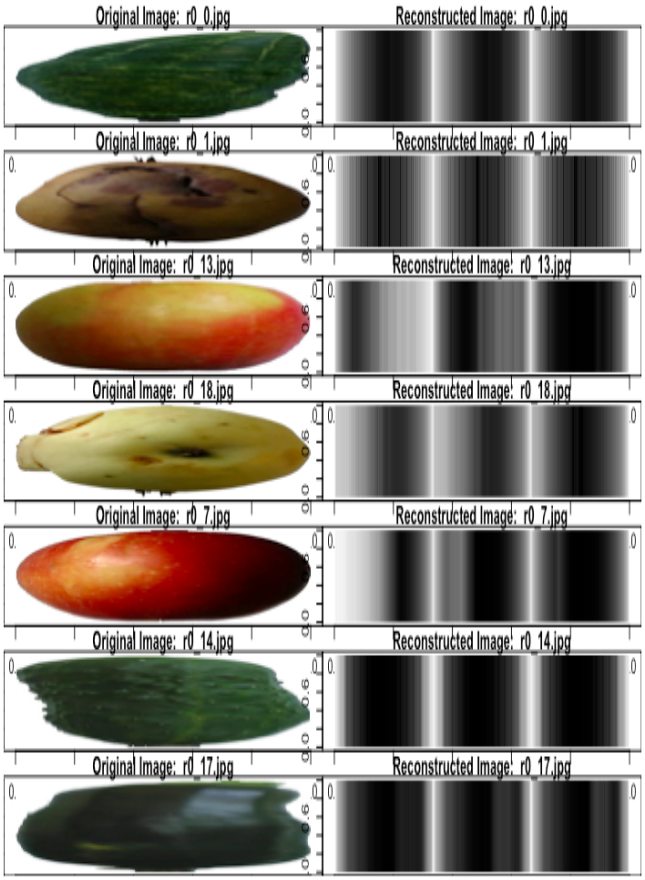

Figure 5: Original and reconstructed images for seven images.

Since this is successful, we will apply PCA to all of our images by the following code (Figure 6). One concern is this process can take a very long time.

```r
```{r}
library(jpeg)
library(grid)
library(dplyr)
library(pracma)

image_dir <- "/Users/guosiwei/Desktop/All images/"

read_and_flatten <- function(image_file) {
  img <- readJPEG(image_file)
  img_flattened <- as.vector(img)
  return(img_flattened)
}

apply_pca <- function(image_files) {
  images <- lapply(image_files, read_and_flatten)
  images_combined <- do.call(rbind, images)
  pca_result <- prcomp(images_combined)
  return(pca_result)
}

image_folders <- list.dirs(image_dir, full.names = TRUE, recursive = FALSE)

for (folder in image_folders) {
  image_files <- list.files(folder, full.names = TRUE)
  if (length(image_files) > 0) {
    par(mfrow = c(3, 5), mar = c(0, 0, 0, 0))
    for (i in 1:min(15, length(image_files))) {
      img <- readJPEG(image_files[i])
      plot(1, type = 'n', xlab = '', ylab = '', xlim = c(0, 1), ylim = c(0, 1))
      rasterImage(img, 0, 0, 1, 1)
    }
    par(mfrow = c(1, 1))

    pca_result <- apply_pca(image_files)

    print(summary(pca_result))
  } else {
    cat("No images found in", folder, "\n")
  }
}
```
```

Figure 6: Sample R Code for PCA Analysis

Preliminary research ideas and future plan:

General Research Ideas:

1. Collecting fruit images.

```
train_path = '/Users/66smacbookpro/Desktop/archive (3)/Training/'
test_path = '/Users/66smacbookpro/Desktop/archive (3)/Test/'
```

```
fig = plt.figure(figsize=(20,10))

for i in range(10):
    ax = fig.add_subplot(6,5,i+1, xticks=[], yticks=[])
    plt.title(df_train_nuniques['fruits_name'][i])
    plt.axis('off')
    file = os.listdir(train_path + df_train_nuniques['fruits_name'][i])

    img_name = file[0]
    ax.imshow(image.load_img(train_path + df_train_nuniques['fruits_name'][i] + '/' + img_name))
```

Figure 7: Fruit images preview

2. Image preprocessing: Different images have different sizes and angles. As a result, we apply preprocessing techniques including resizing images to make sure all images have a uniform size, normalizing the pixel values to a desired range (0-

1), and applying data augmentation including grey scaling, reflection, and rotation of the image to prepare for further machine learning applications.

```
img=cv2.imread(train_path+"Plum 3/r_91_100.jpg")
plt.figure(figsize=(2,2))
plt.imshow(img)
plt.axis("off")
plt.show()
img_normalized = cv2.normalize(img, None, 0, 1.0, cv2.NORM_MINMAX, dtype=cv2.CV_32F)
print("Image data before Normalize:\n", img)

ret,thresh = cv2.threshold(img,140,255,cv2.THRESH_BINARY)
print("Image data after Thresholding:\n", thresh)

img_normalized = cv2.normalize(thresh, None, 0, 1.0, cv2.NORM_MINMAX, dtype=cv2.CV_32F)

cv2.imshow('Normalized Image', img_normalized)
cv2.waitKey(0)
cv2.destroyAllWindows()
print("Image data after Normalize:\n", img_normalized)

img2=cv2.imread(train_path+"Plum 3/r_91_100.jpg")
plt.figure(figsize=(2,2))
plt.imshow(img)
plt.axis("off")
plt.show()
```

Figure 8: Original image of Plum 3

Figure 9: Normalized image of Plum 3

3. Classification: Transfer Learning and Convolutional Neural Networks (CNNs) are planned to be used for classification. Because of the limitation of images, we apply Transfer Learning to make our model more accurate and lower generalization error.

We will use the pre-trained CNNs model such as ResNet and MobileNet to train the dataset we use, and continuously adjust the model parameters through the backpropagation algorithm so that the model can better extract features in the image and perform classification.

During the training process, we can also improve the performance of the model by adjusting hyperparameters, transfer learning and other means. After the training is completed, we will use the test dataset to test the model and evaluate the performance of the model to ensure that our CNNs model has high accuracy and practicality in

fruit classification prediction and grade recognition.

```
model <- keras_model_sequential()


model %>%
  layer_conv_2d(filter = 16, kernel_size = c(2,2), padding="same", input_shape = c(img_width, img_height, channels)) %>%
  layer_activation("relu") %>%
  layer_batch_normalization() %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%

  layer_conv_2d(filter = 32, kernel_size = c(2,2)) %>%
  layer_activation("relu") %>%
  layer_batch_normalization() %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%


  layer_conv_2d(filter = 64, kernel_size = c(2,2),padding="same") %>%
  layer_activation("relu") %>%
  layer_batch_normalization() %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%

  layer_conv_2d(filter = 128, kernel_size = c(2,2),padding="same") %>%
  layer_activation("relu") %>%
  layer_batch_normalization() %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%
  layer_dropout(0.5) %>%
  layer_flatten() %>%
  layer_dense(150) %>%
  layer_activation("relu") %>%
  layer_dropout(0.5) %>%

  layer_dense(output_n) %>%
  layer_activation("softmax")

model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = "adam",
  metrics = "accuracy"
)
summary(model)
```

```
Model: "sequential_3"
_________________________________________________________________________________________
 Layer (type)                        Output Shape                      Param #     Trainable
=========================================================================================
 conv2d_9 (Conv2D)                   (None, 100, 100, 16)              208         Y
 activation_14 (Activation)          (None, 100, 100, 16)              0           Y
 batch_normalization_8 (BatchNorma   (None, 100, 100, 16)              64          Y
 lization)
 max_pooling2d_8 (MaxPooling2D)      (None, 50, 50, 16)                0           Y
 conv2d_8 (Conv2D)                   (None, 49, 49, 32)                2080        Y
 activation_13 (Activation)          (None, 49, 49, 32)                0           Y
 batch_normalization_7 (BatchNorma   (None, 49, 49, 32)                128         Y
 lization)
 max_pooling2d_7 (MaxPooling2D)      (None, 24, 24, 32)                0           Y
 conv2d_7 (Conv2D)                   (None, 24, 24, 64)                8256        Y
 activation_12 (Activation)          (None, 24, 24, 64)                0           Y
 batch_normalization_6 (BatchNorma   (None, 24, 24, 64)                256         Y
 lization)
 max_pooling2d_6 (MaxPooling2D)      (None, 12, 12, 64)                0           Y
 conv2d_6 (Conv2D)                   (None, 12, 12, 128)               32896       Y
 activation_11 (Activation)          (None, 12, 12, 128)               0           Y
 batch_normalization_5 (BatchNorma   (None, 12, 12, 128)               512         Y
 lization)
 max_pooling2d_5 (MaxPooling2D)      (None, 6, 6, 128)                 0           Y
 dropout_5 (Dropout)                 (None, 6, 6, 128)                 0           Y
 flatten_2 (Flatten)                 (None, 4608)                      0           Y
 dense_5 (Dense)                     (None, 150)                       691350      Y
 activation_10 (Activation)          (None, 150)                       0           Y
 dropout_4 (Dropout)                 (None, 150)                       0           Y
 dense_4 (Dense)                     (None, 131)                       19781       Y
 activation_9 (Activation)           (None, 131)                       0           Y
=========================================================================================
Total params: 755531 (2.88 MB)
Trainable params: 755051 (2.88 MB)
Non-trainable params: 480 (1.88 KB)
_________________________________________________________________________________________
```

Figure 10 & 11: Model Architecture Visualization

There are several possible extensions that we can have on our research. For example, all the image files in our dataset have white backgrounds. In reality, there must be some

images with backgrounds of different colors. This is exactly one extension we can have to try using our research model (CNN) and outcomes to analyze images with different backgrounds and see if they still work. Our research might also have another direction in the future by considering analyzing the images of animals instead of fruits. This can be a more interesting direction because there are way more animal species than of fruits.

Implication:

We hope that through our research, we can more accurately identify specific image types and situations based on nuances, and can monitor them at any time according to changes. Applying this model to daily life is especially convenient for things that are perishable and difficult to store, such as fruits and vegetables.This can help customers easily distinguish fruits with similar characteristics, and it can also help sellers create labels more effectively and reduce confusion.

With our image classification, we can apply it to places where fruits are sold, such as grocery stores and supermarkets. The machine analyzes the image to determine the category, which can simplify the overall operation of the supermarket and present customers with better and more accurate visual effects.

Acknowledgement:

We extend our gratitude to the contributors of the dataset obtained from https://www.kaggle.com/datasets/moltean/fruits, which is licensed under CC BY-SA 4.0. This license grants researchers the freedom to share (copy and redistribute the material in any medium or format for any purpose, even commercially) and adapt (remix, transform, and build upon the material for any purpose, even commercially). We acknowledge and appreciate the generosity of the licensors in providing this valuable resource for our research.

Citations:

1. Mihai Oltean. (n.d.). Fruits Image Dataset. Kaggle. https://www.kaggle.com/datasets/moltean/fruits

2. Sanghvi, K. (2023, April 21). *Image classification techniques*. Medium. https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac

3. Han, D., Liu, Q., & Fan, W. (2018). A new image classification method using CNN transfer learning and web data augmentation. *Expert Systems with Applications*, *95*, 43–56. https://doi.org/10.1016/j.eswa.2017.11.028

4. Naik, Sapan & Patel, Bankim. (2017). Machine Vision based Fruit Classification and Grading - A Review. International Journal of Computer Applications. 170. 22-34. 10.5120/ijca2017914937.

5. M. S. Hossain, M. Al-Hammadi and G. Muhammad, "Automatic Fruit Classification Using Deep Learning for Industrial Applications," in IEEE Transactions on Industrial Informatics, vol. 15, no. 2, pp. 1027-1034, Feb. 2019, doi: 10.1109/TII.2018.2875149.

6. Miao, Z., Gaynor, K.M., Wang, J. *et al.* Insights and approaches using deep learning to classify wildlife. *Sci Rep* **9**, 8137 (2019). https://doi.org/10.1038/s41598-019-44565-w