Quality Control of Fruit by Type and Freshness Using Machine Learning Techniques

Mario Canche Mex, Brayan Fonseca Carrillo, Alan Herrera Tuz, and Alexis Hoil Sosa,

***Abstract*—**

***Index Terms*—Machine Learning, Supervised Learning, Unsupervised Learning, Reinforcement Learning, Fruit Freshness, Fruit Type, Quality Control.**

I. Introduction

THIS study includes a thorough investigation into the use of supervised, unsupervised, and reinforcement learning approaches for fruit quality management, with an emphasis on apples, bananas, and oranges in both fresh and rotting circumstances. Using cutting-edge machine learning approaches and computational robotics, we solve the issues of fruit classification and quality assessment. In the supervised learning portion, a convolutional neural network (CNN) with a predictive model capable of distinguishing between the six separate classes of fruits was created using TensorFlow and Keras. The dataset, which included over 10,000 photos, was separated into three sets: training, validation, and test, with each set further broken into six folders corresponding to the various fruit groups. To improve model generalization, data augmentation approaches were used, which were done using the OS and PIL Python packages. Accuracy assessments on training, validation, and test datasets, as well as predictions on fresh data, were included in the evaluation, along with confidence percentage measures. In the network layers, the architecture included ReLU and sigmoidal activation functions. The same dataset was used without labels in the unsupervised learning phase, with the purpose of clustering the photos into six unique groups. utilizing leaky ReLU activation functions, a predictor model was created utilizing data augmentation techniques similar to the supervised learning phase. The model's ability to group photos was tested using k-means clustering, and the results were visually reviewed using Matplotlib's random image presentations. The third section introduces a reinforcement learning model specifically tailored for a simulation environment, ROS-Gazebo. A detection band was used to determine fruit type and quality in a virtual situation, following which a robotic arm applied a force 'X' to each fruit. The goal was to teach the robotic arm the best method for choosing fruits depending on their qualities. To solve the numerous issues connected with automated fruit quality management, this multidisciplinary work connects the disciplines of computational robotics, machine learning, and artificial intelligence. The next parts go into the methodology, findings, and comments, providing a comprehensive understanding of the approaches and their implications in agricultural automation and robotics.

II. Main Problem

The fundamental goal of this project is to handle the hard task of automating fruit quality management by combining modern machine learning techniques and computational robotics. The underlying issue can be divided into numerous essential components, each requiring careful thought and imaginative solutions:

A. *Supervised Learning for Fruit Classification*

The job entails creating a strong convolutional neural network (CNN) architecture using TensorFlow and Keras. The algorithm must accurately categorize fruits into six unique groups, including fresh and rotting apples, bananas, and oranges. The optimization of hyperparameters, network configurations, and the model's generalization capacity to accommodate new, unknown data is critical.

B. *Unsupervised Learning for Fruit Clustering*

Unsupervised learning techniques are used to categorize fruits into six distinct classes, removing the necessity for labeled data. Using clustering techniques like k-means necessitates a thorough examination of the model's ability to classify fruits based on innate visual qualities. Visual inspections and relevant clustering metrics are used in the assessment.

C. *Reinforcement Learning for Robotic Arm Control*

It is anticipated that a reinforcement learning model might be built in a simulated environment enabled by ROS-Gazebo to enable a robotic arm to select fruits optimally. The proposed task entails developing a hypothetical effective reward system to lead the learning process toward the anticipated desired outcome of applying an ideal force 'X' to each fruit based on its type and quality. The success of the hypothetical learning process would be measured in terms of convergence and efficiency, assuming that these are important criteria in this context.

D. Integration of Machine Learning and Robotics

It is critical to have seamless integration between machine learning models and the robotic arm within the ROS-Gazebo simulation environment. To create a cohesive and successful connection, compatibility issues between ROS-Gazebo and TensorFlow/Keras must be handled. The taught robotic arm control algorithms' real-world adaptability and efficacy contribute even more to the problem's full resolution.

III. Proposed solution

A. Supervised Solution

The proposed solution for the supervised learning approach in the provided code involves the development and training of a convolutional neural network (CNN) for the classification of fruits into distinct categories based on their type (apples, bananas, oranges) and quality (fresh or rotten). The solution comprises the following key steps:

- **Data Organization:** A meticulously structured dataset is created, encompassing separate directories for training, validation, and test sets, each further categorized into subdirectories representing the different types and conditions of fruits.
- **Data Augmentation:**Leveraging the ImageDataGenerator from the Keras library, data augmentation techniques are applied during training to enhance the model's ability to generalize to diverse scenarios. These augmentation methods include rotation, width and height shifts, shear, zoom, and flips, contributing to increased dataset variability.
- **Model Architecture:** A convolutional neural network (CNN) is constructed using TensorFlow/Keras, featuring multiple convolutional layers, max-pooling layers, and dense layers. The architecture is designed to capture hierarchical features within the input images and culminates in a softmax activation function for multi-class classification.
- **Training and Evaluation:** The model is trained using the training dataset, with an emphasis on achieving a high level of accuracy. During training, a custom callback is implemented to halt the training process upon reaching a predefined accuracy threshold, in this case, 98 percent. The model's performance is evaluated on both the training and validation sets.
- **Test Data Evaluation:** The trained model undergoes evaluation using an independent test dataset to assess its generalization capability and accuracy on previously unseen data.
- **Prediction on New Images:**The solution extends to the prediction of fruit classes for new images uploaded by the user. The model is loaded, and predictions are made on these images, providing insights into the model's performance and its ability to classify fruits in real-world scenarios.

B. Unsupervised solution

- **Data Preparation** The solution commences with meticulous data preparation. The fruit dataset is systematically organized into distinct directories for training, validation, and testing. This establishes a structured foundation for subsequent model training and evaluation. Furthermore, image augmentation techniques are applied to the training dataset using TensorFlow Keras's ImageDataGenerator. This step is crucial for enhancing the model's capacity to generalize by introducing variations in the training images, thereby improving its ability to recognize diverse patterns.
- **DataFrame Creation** A dataframe is generated to simplify the link between filenames and unique labels. This dataframe contains important metadata about each image and serves as a structured representation of the dataset. Each image is given a distinct label, bridging the gap between filenames and distinct IDs. This standardized approach makes it easier to generate an unsupervised learning generator later on.
- **Unsupervised Learning generator** Leveraging the dataframe, an unsupervised learning generator is instantiated using the flow-from-dataframe function from TensorFlow Keras. This generator becomes the conduit for producing batches of augmented images devoid of explicit labels, a fundamental requirement for unsupervised learning endeavors. The generator is configured to shuffle the data to prevent any bias during model training, and the target size is set to ensure consistency in image dimensions.
- **Model for Unsupervised learning** A convolutional neural network (CNN) serves as the cornerstone for unsupervised learning. This model is designed to learn a latent space representation of the fruit images, capturing intricate features without relying on predefined classes. The CNN architecture encompasses convolutional layers, Leaky ReLU activation functions, max-pooling layers, and dense layers. The model is optimized for learning latent structures within the fruit dataset.
- **Clustering using KMeans**
  The latent space representations acquired from the CNN are pivotal for subsequent unsupervised learning. These representations encapsulate learned features that are crucial for identifying patterns and similarities among images. KMeans clustering is applied to these latent space representations, serving as the unsupervised mechanism for grouping images into clusters. This process is foundational for revealing inherent structures within the dataset, uncovering relationships that may not be immediately apparent.
- **Outcome and significance**
  Image augmentation, CNN-based latent space learning, and KMeans clustering work together to create a comprehensive unsupervised learning solution. The model discovers patterns and structures in the fruit dataset on its own, delivering significant insights without the need for

manual labeling. The importance of this method resides in its capacity to uncover latent features autonomously, making it usable in cases where labeled data is scarce or absent. The given unsupervised learning methodology contributes to the broader landscape of artificial intelligence by providing a data-driven strategy to interpreting difficult datasets.

C. Reinforcement solution

”In the realm of robotics, there are numerous ways to integrate reinforcement learning. In our context, the approach to incorporating this topic involves utilizing a robotic arm capable of modulating its force based on the type and condition of fruits. The primary function of this arm is to remove all spoiled fruits from the conveyor belt. To achieve this, the robot would be trained using tactile sensors to estimate the object’s rigidity. Subsequently, it can dynamically adjust the necessary force to handle the spoiled fruit without causing damage.”

IV. Description of the Data Used

The dataset used, as mentioned above, was one that contained 3 different types of fruits, among which were apples, oranges and bananas, each in its fresh and rotten version, thus having a total of 6 possible classes, fresh apples, fresh bananas. , fresh oranges, rotten apples, rotten bananas and rotten oranges, the dataset in total has a number of 13,600 images to perform different operations with our models, in the case of supervised learning the dataset was divided into 3 main folders among which were training with 60 percent of the possible information of the dataset, the validation with 20 percent of the dataset and finally the test folder with also 20 percent of the information. Each of these folders was subdivided into 6 folders with the possible classes mentioned which served as labels in unsupervised learning.

SRIRAM REDDY KALLURI · UPDATED 5 YEARS AGO

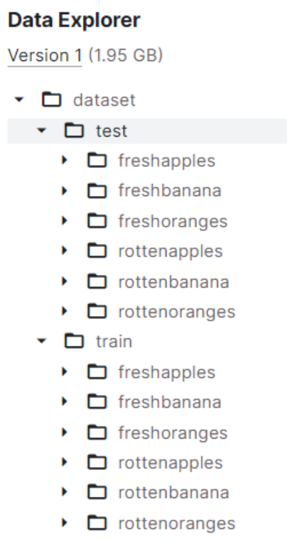
299

Fruits fresh and rotten for classification

Apples Oranges Bananas

Fig. 1. Page of dataset source

For unsupervised learning, the decision was made to eliminate the class folders since they were not necessary, and test and validation folders were also created with only 10 percent of the information, which were not used in the future since unsupervised learning works differently. and the architecture of the model turned out to be different, in this version of the dataset there were also a total of 13,600 images.



Data Explorer

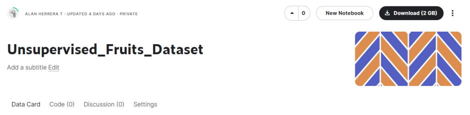
Version 1 (1.95 GB)

- dataset
  - test
    - freshapples
    - freshbanana
    - freshoranges
    - rottenapples
    - rottenbanana
    - rottenoranges
  - train
    - freshapples
    - freshbanana
    - freshoranges
    - rottenapples
    - rottenbanana
    - rottenoranges

Fig. 2. Supervised dataset configuration

To work on the dataset, the Kaggle APIs were used in Google Collab (the environment in which the models were programmed), the first dataset used was found on the Internet and a split size was simply applied to it and management of Google Collab’s internal folders was used. from the OS library and Google files, the dataset had to be divided by code since it was initially poorly distributed, data augmentation was also applied using the PIL library as described above. For

the following models, the dataset was combined by hand as mentioned and that version was uploaded to kaggle in order to be able to download it using the API, the data was managed internally in collab using the OS library and data was applied. aumegtention using the PIL library as described.



ALAN HERRERA T · UPDATED 4 DAYS AGO · PRIVATE

0 New Notebook Download (2 GB)

Unsupervised_Fruits_Dataset

Add a subtitle Edit

Data Card Code (0) Discussion (0) Settings

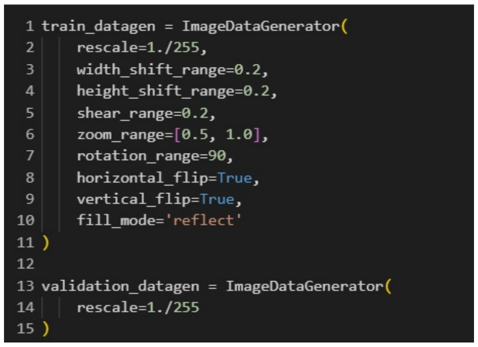
Fig. 3. Unsupervised dataset

V. Description of the Methodology Used

A. Unsupervised methodology

- **Data Colletcion and Organization** The process begins with meticulously organizing a dataset, which is accomplished by creating directories for training, validation, and test sets. Subdirectories inside each set contain a variety of fruit varieties and circumstances. The makedir function is used to build this organizational structure, which assures the right arrangement of fresh and decaying fruits for apples, bananas, and oranges.
- **Data Augmentation** To enhance model robustness, data augmentation techniques are implemented using the ImageDataGenerator from Keras. Augmentations include rotation, shifts in width and height, shear, zoom, and flips. This is demonstrated in the train-datagen setup,

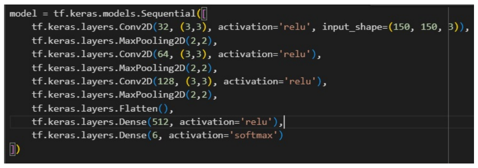
influencing the generation of augmented images during training.



```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=[0.5, 1.0],
    rotation_range=90,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='reflect'
)

validation_datagen = ImageDataGenerator(
    rescale=1./255
)
```

Fig. 4. Data augmentation

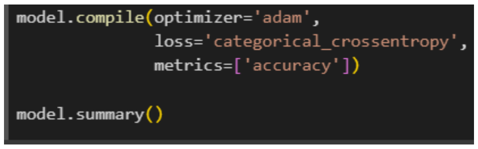
- **Convolutional Neural Network (CNN) Architecture** The CNN model is architecturally designed using TensorFlow/Keras. Convolutional layers capture hierarchical features, max-pooling layers reduce dimensionality, and dense layers enable classification. The model concludes with a softmax activation function for multi-class prediction.



```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(6, activation='softmax')
])
```

Fig. 5. CNN architecture

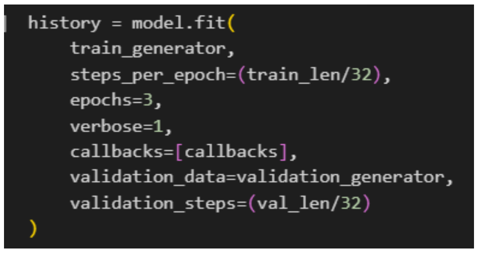
- **Model Compilation and Summary** The CNN model is compiled with the Adam optimizer and categorical cross-entropy loss. Model summary provides an overview of layer configurations and parameters, facilitating an in-depth understanding of the model's architecture.



```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

Fig. 6. Model compilation and summary

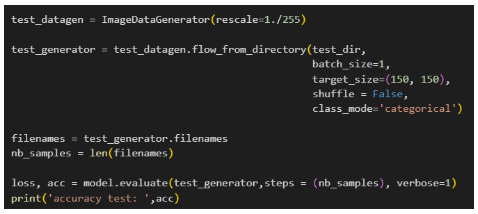
- **Training and Custom Callback**
The model undergoes training with the fit function, utilizing the train-generator and validation-generator. A custom callback (myCallback) halts training upon reaching a specified accuracy threshold (98 percent).



```
history = model.fit(
    train_generator,
    steps_per_epoch=(train_len/32),
    epochs=3,
    verbose=1,
    callbacks=[callbacks],
    validation_data=validation_generator,
    validation_steps=(val_len/32)
)
```

Fig. 7. Training and custom callback

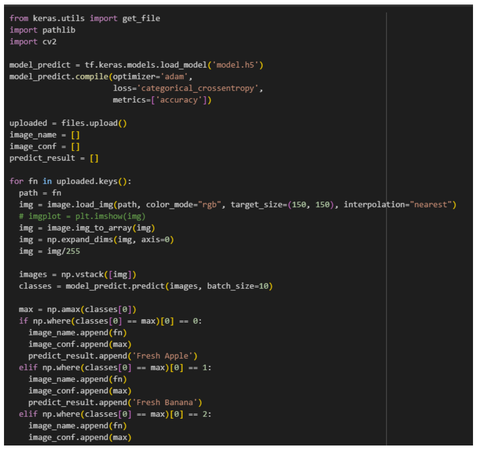
- **Evaluation on Training, Validation and Test sets** The trained model is carefully tested on training, validation, and test datasets, allowing us to measure its performance over a wide range of scenarios. In assessing the model's categorization capabilities, evaluation criteria such as accuracy are critical.



```
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(test_dir,
                                                  batch_size=1,
                                                  target_size=(150, 150),
                                                  shuffle = False,
                                                  class_mode='categorical')


filenames = test_generator.filenames
nb_samples = len(filenames)

loss, acc = model.evaluate(test_generator,steps = (nb_samples), verbose=1)
print('accuracy test: ',acc)
```

Fig. 8. Evaluation on test

- **Prediction on New images** The model is applied to predict classes for new images uploaded by the user. These predictions, along with confidence scores, are showcased using Matplotlib, offering a practical demonstration of the model's utility in real-world scenarios.



```
from keras.utils import get_file
import pathlib
import cv2

model_predict = tf.keras.models.load_model('model.h5')
model_predict.compile(optimizer='adam',
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])

uploaded = files.upload()
image_name = []
image_conf = []
predict_result = []

for fn in uploaded.keys():
  path = fn
  img = image.load_img(path, color_mode="rgb", target_size=(150, 150), interpolation="nearest")
  # imgplot = plt.imshow(img)
  img = image.img_to_array(img)
  img = np.expand_dims(img, axis=0)
  img = img/255

  images = np.vstack([img])
  classes = model_predict.predict(images, batch_size=10)

  max = np.amax(classes[0])
  if np.where(classes[0] == max)[0] == 0:
    image_name.append(fn)
    image_conf.append(max)
    predict_result.append('Fresh Apple')
  elif np.where(classes[0] == max)[0] == 1:
    image_name.append(fn)
    image_conf.append(max)
    predict_result.append('Fresh Banana')
  elif np.where(classes[0] == max)[0] == 2:
    image_name.append(fn)
    image_conf.append(max)
```

Fig. 9. Prediction on new images

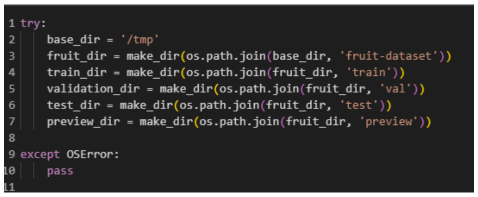
B. Supervised Methodology

The methodology for the unsupervised learning model consists of a sequence of well-defined phases that allow the machine to learn and extract significant patterns from the fruit dataset independently. Here is a thorough explanation of the methodology:

- **Data organization and augmentation**

The dataset is systematically organized into training, validation, and test directories to facilitate model training and evaluation. Image augmentation is applied to the training dataset using TensorFlow Keras's ImageDataGenerator. This step introduces variations in the training images, enhancing the model's generalization capabilities.

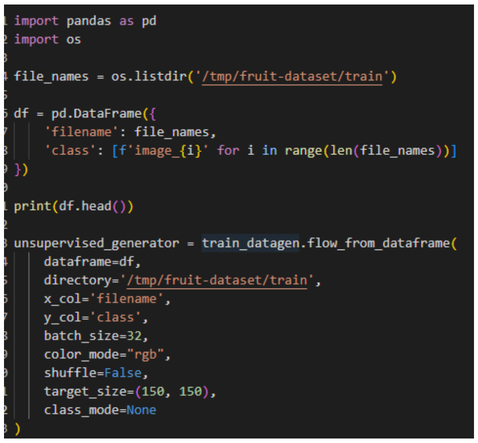
- **DataFrame construction** To provide a structured representation of the dataset, a dataframe is developed to correlate filenames with unique labels. Each image is given a unique label, bridging the gap between filenames and individual IDs.



```
try:
    base_dir = '/tmp'
    fruit_dir = make_dir(os.path.join(base_dir, 'fruit-dataset'))
    train_dir = make_dir(os.path.join(fruit_dir, 'train'))
    validation_dir = make_dir(os.path.join(fruit_dir, 'val'))
    test_dir = make_dir(os.path.join(fruit_dir, 'test'))
    preview_dir = make_dir(os.path.join(fruit_dir, 'preview'))

except OSError:
    pass

```

Fig. 10. Folders creation for unsupervised model

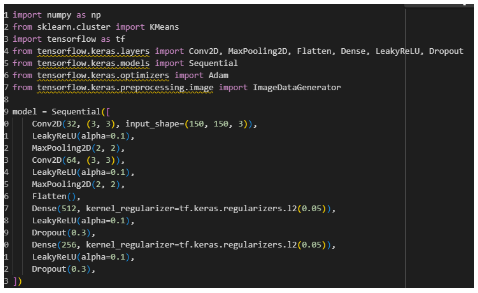
- **Unsupervised learning generator** The flow-from-dataframe function in TensorFlow Keras generates an unsupervised learning generator using the dataframe. This generator is set up to generate batches of augmented photos without explicit labels, which is an important part of unsupervised learning.



```
import pandas as pd
import os

file_names = os.listdir('/tmp/fruit-dataset/train')

df = pd.DataFrame({
    'filename': file_names,
    'class': [f'image_{i}' for i in range(len(file_names))]
})

print(df.head())

unsupervised_generator = train_datagen.flow_from_dataframe(
    dataframe=df,
    directory='/tmp/fruit-dataset/train',
    x_col='filename',
    y_col='class',
    batch_size=32,
    color_mode="rgb",
    shuffle=False,
    target_size=(150, 150),
    class_mode=None
)
```

Fig. 11. Unsupervised learning generator

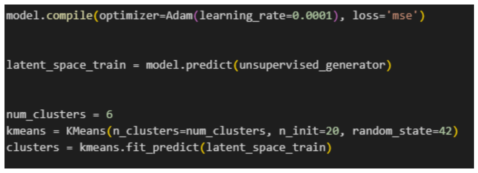
- **Convolutional Neural Network Architecture** A CNN is the fundamental architecture for unsupervised learning. This model is intended to learn a latent space representation of fruit photos in order to capture nuanced aspects without relying on predefined classes. Convolutional layers, Leaky ReLU activation functions, max-pooling layers, and thick layers compose the CNN architecture. This architecture is specifically designed to extract and represent hierarchical characteristics in a dataset. The CNN is trained on the augmented training dataset to learn a latent space representation of the images. The latent space encapsulates abstract features that characterize the intrinsic patterns within the dataset.



```
import numpy as np
from sklearn.cluster import KMeans
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, LeakyReLU, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator

model = Sequential([
    Conv2D(32, (3, 3), input_shape=(150, 150, 3)),
    LeakyReLU(alpha=0.1),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3)),
    LeakyReLU(alpha=0.1),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(512, kernel_regularizer=tf.keras.regularizers.l2(0.05)),
    LeakyReLU(alpha=0.1),
    Dropout(0.3),
    Dense(256, kernel_regularizer=tf.keras.regularizers.l2(0.05)),
    LeakyReLU(alpha=0.1),
    Dropout(0.3),
])
```

Fig. 12. CNN model for unsupervised learning

- **Kmeans Clustering** The CNN's latent space representations are sent into the KMeans clustering algorithm. KMeans clustering divides photos into clusters automatically based on similarities in their latent space representations. The number of clusters is often predetermined, representing the dataset's supposed natural groupings.



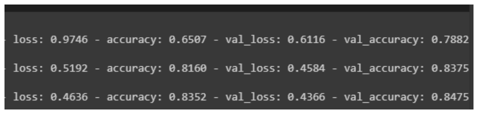
```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='mse')


latent_space_train = model.predict(unsupervised_generator)


num_clusters = 6
kmeans = KMeans(n_clusters=num_clusters, n_init=20, random_state=42)
clusters = kmeans.fit_predict(latent_space_train)
```

Fig. 13. Clustering model

VI. Evaluation Method Used

A. Supervised Evaluation

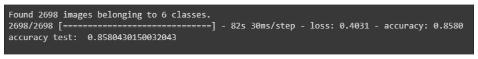
This study's assessment methodology takes a systematic approach to assessing the performance of the convolutional neural network (CNN) model developed for fruit classification. The evaluation spans numerous aspects, offering a thorough insight of the model's effectiveness by leveraging the capabilities of TensorFlow Keras. The model's efficacy in learning from the training dataset is measured during the training phase. The fit function, a key component of TensorFlow Keras, aids in this process. The training accuracy is an important parameter since it reflects the model's capacity to adapt to the complexities of the training data.



```
loss: 0.9746 - accuracy: 0.6507 - val_loss: 0.6116 - val_accuracy: 0.7882

loss: 0.5192 - accuracy: 0.8160 - val_loss: 0.4584 - val_accuracy: 0.8375

loss: 0.4636 - accuracy: 0.8352 - val_loss: 0.4366 - val_accuracy: 0.8475
```

Fig. 14. Results on training

The validation procedure is simplified using TensorFlow Keras, which allows the model's generalization capabilities to be tested in real time. The flow-from-directory function assists in the generation of validation data batches, allowing validation accuracy to be calculated. This statistic is an important indicator of the model's capacity to apply its knowledge to previously encountered cases.



```
Found 2698 images belonging to 6 classes.
2698/2698 [==============================] - 82s 30ms/step - loss: 0.4031 - accuracy: 0.8580
accuracy test:  0.8580430150032043
```

Fig. 15. Results on test

The ultimate test of the model's proficiency is conducted on a dedicated test dataset, carefully isolated from the training and validation phases. The evaluate function in TensorFlow Keras calculates the loss and accuracy on this dataset. This step ensures an unbiased assessment of the model's performance on novel, unencountered data. TensorFlow Keras supports the use of custom callbacks to improve training control. In this case, a custom callback (myCallback) is used to track training accuracy. Training is ended when the accuracy exceeds a predefined threshold (98 percent). TensorFlow Keras's early termination method reduces overfitting and optimizes training efficiency.

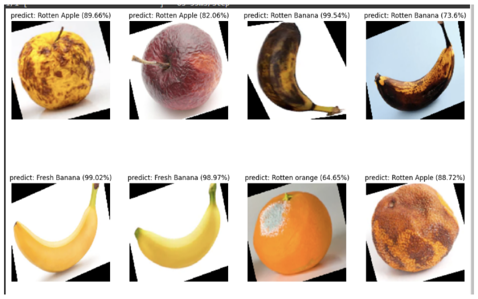
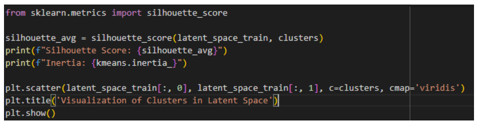

predict: Rotten Apple (89.66%)
predict: Rotten Apple (82.06%)
predict: Rotten Banana (99.54%)
predict: Rotten Banana (73.6%)
predict: Fresh Banana (99.02%)
predict: Fresh Banana (98.97%)
predict: Rotten orange (64.65%)
predict: Rotten Apple (88.72%)

Fig. 16. Results on random images

To complement quantitative metrics, the TensorFlow Keras integration facilitates the visual inspection of model predictions on new images uploaded by the user. Utilizing Matplotlib, this qualitative assessment offers insights into the model's real-world applicability.

B. Unsupervised Evaluation

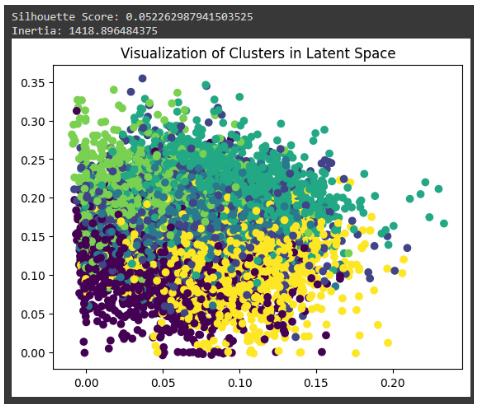
Evaluating the efficacy of an unsupervised learning model can be difficult because typical measurements such as accuracy, precision, or recall are ineffective in the absence of ground truth labels. In this case we apply an evaluation metric that is described below, in which the results of the already trained model are used.



```python
from sklearn.metrics import silhouette_score

silhouette_avg = silhouette_score(latent_space_train, clusters)
print(f"Silhouette Score: {silhouette_avg}")
print(f"Inertia: {kmeans.inertia_}")

plt.scatter(latent_space_train[:, 0], latent_space_train[:, 1], c=clusters, cmap='viridis')
plt.title('Visualization of Clusters in Latent Space')
plt.show()
```

Fig. 17. Silhouette score code

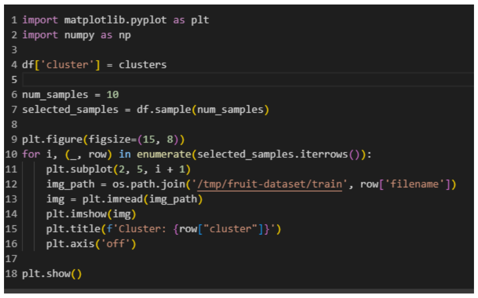
The Silhouette Score, calculated with the sklearn.metrics silhouette-score function, provides a quantitative measure of the separation and coherence of the clusters created in the latent space. A higher Silhouette Score suggests that the clusters are well-defined and distinct. In addition, the assessment of the model's inertia, acquired via kmeans.inertia-, provides insights into the clusters' compactness. Lower inertia levels are typically associated with more cohesive clusters.



```
Silhouette Score: 0.052262987941503525
Inertia: 1418.896484375
```

Fig. 18. Visualization of Clusters in Latent Space

The following code segment helps visual exploration and analysis of unsupervised learning model clustering outcomes. The algorithm randomly selects a subset of samples for presentation after assigning cluster labels to the original DataFrame containing information about fruit photos. The resulting visualization, presented on a 2x5 grid, displays photos alongside their associated cluster names. This graphic depiction is used to provide a qualitative assessment of the clustering results. The algorithm allows direct observation of the model's grouping decisions by linking images with their assigned clusters, assisting in the understanding of the latent space clustering.



```python
import matplotlib.pyplot as plt
import numpy as np

df['cluster'] = clusters

num_samples = 10
selected_samples = df.sample(num_samples)

plt.figure(figsize=(15, 8))
for i, (_, row) in enumerate(selected_samples.iterrows()):
    plt.subplot(2, 5, i + 1)
    img_path = os.path.join('/tmp/fruit-dataset/train', row['filename'])
    img = plt.imread(img_path)
    plt.imshow(img)
    plt.title(f'Cluster: {row["cluster"]}')
    plt.axis('off')

plt.show()
```

Fig. 19. Code to have a visualization of the clustering results

Such representations are extremely useful for scholars and practitioners looking to understand the patterns collected by the unsupervised learning model, enabling a more intuitive grasp of how unique clusters relate to the fundamental qualities of the fruit photos dataset. This approach is consistent with existing data exploration practices, allowing researchers to acquire insights into the clustering structure and evaluate the model's capacity to detect important patterns within the dataset.

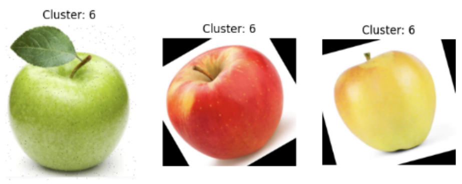


Fig. 20. Some results of the clusterization

VII. Challenges faced during the development of the project

During the development of the task we faced different challenges, such as finding a dataset that would represent what was proposed at the beginning of the subject, in which we proposed a classification of fruits by type and quality, which as we can see throughout this report was achieved, we also faced a total lack of knowledge of various aspects of the topics such as the management of data, folders and files of the system that we had to learn from scratch, in turn when using images to carry out our work we had to using convolutional neural networks which was also a totally new topic for us, in the supervised and unsupervised learning part those were our barriers when working on the project, since we faced all this completely alone without almost any type of guide.

Due to time constraints, the implementation of the core idea of reinforcement learning could not be realized in the planned project. Most of the components had to be designed and programmed, since the simulation of such situations is best achieved in ROS. In ROS, a person can simulate a context to create the desired situation, including physics. However, achieving this requires a significant amount of time, encompassing the need to design, program and implement these simulations. It is a task that is almost done tailored for a person pursuing a master's degree in ROS, at the same time we have zero knowledge of simulation environments and the compatibility that these may present with the data or packages and libraries that we planned to use.

The biggest problem presented in conclusion was the lack of time to work on all aspects of the project, since our time was divided and it was difficult to dedicate the time to learn as well as implement.

VIII. Conclusion and future steps

A complex methodology was used to address the discerning of fruit types and qualities in this complete examination of fruit classification within the realms of supervised and unsupervised learning paradigms. A Convolutional Neural Network (CNN) was methodically developed in the supervised learning domain, leveraging the TensorFlow and Keras frameworks, to categorize apples into six separate classes based on both type and freshness. To improve model generality, extensive data augmentation approaches were used, resulting in a rigorous evaluation procedure that encompassed training, validation, and testing phases. Using a well curated dataset of over 10,000 photos, the model demonstrated noteworthy accuracy in identifying fruit classifications, demonstrating its use in supervised learning applications. In addition, the unsupervised learning component investigated clustering fruit photos in a latent space. The use of a K-means clustering technique on the latent space representation of images generated by a convolutional neural network aided in the discovery of intrinsic patterns and groups in the dataset. The evaluation indicators, such as the Silhouette Score and cluster visualization, provided quantitative and qualitative insights into the unsupervised learning model's effectiveness. The combination of supervised and unsupervised learning algorithms provides a comprehensive view of fruit categorization, responding to scenarios involving labeled and unlabeled data, respectively. This methodological duality increases the entire framework's versatility and applicability, demonstrating its potential for effective fruit classification in a variety of scenarios. Furthermore, the stringent evaluation measures used in both paradigms help to provide a more nuanced knowledge of the models' performance, establishing the framework for informed decision-making in the implementation of machine learning solutions for fruit classification. As future steps, we could undoubtedly improve our unsupervised learning model since it confuses patterns in the images and does not perform completely correct clustering in many types of fruits or classes. At the same time, in the future we could develop the reinforcement learning model which, as mentioned, requires much more time than is currently available.

References

[1] Petar Kormushev, Sylvain Calinon, and D. G. Caldwell, "Reinforcement Learning in Robotics: Applications and Real-World Challenges," Robotics, vol. 2, no. 3, pp. 122–148, Jul. 2013, doi: https://doi.org/10.3390/robotics2030122.

[2] Hadi Yadavari, Vahid Tavakol Aghaei, and Serhat İkizoğlu, "Deep Reinforcement Learning-Based Control of Stewart Platform With Parametric Simulation in ROS and Gazebo," Journal of Mechanisms and Robotics, vol. 15, no. 3, Mar. 2023, doi: https://doi.org/10.1115/1.4056971.

[3] Muhammad Shoaib Akhtar and T. Feng, "Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time," Symmetry, vol. 14, no. 11, pp. 2308–2308, Nov. 2022, doi: https://doi.org/10.3390/sym14112308.

[4] Timothée Lesort, V. Lomonaco, A. Stoian, Davide Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," Information Fusion, vol. 58, pp. 52–68, Jun. 2020, doi: https://doi.org/10.1016/j.inffus.2019.12.004.

[5] L. Fiorini, Gianmaria Mancioppi, F. Semeraro, H. Fujita, and F. Cavallo, "Unsupervised emotional state classification through physiological parameters for social robotics applications," Knowledge Based Systems, vol. 190, pp. 105217–105217, Feb. 2020, doi: https://doi.org/10.1016/j.knosys.2019.105217.