Application of Classification, Clustering and Reinforcement Learning models for Bottle Sorter

Michelle Camara Gonzalez, Eduardo Antonio Flores Arellano, Julio Emanuel Gonzalez Gonzalez, Ricardo Aron Tzuc Zih and Msc. Victor Alejandro Ortiz Santiago.

***Abstract*—In this project, it will be presented a comprehensive solution to efficiently sort and collect PET and glass bottles in industrial and recycling environments. Two fundamental challenges are faced for this: bottle detection and classification using computer vision; the state of art of the pet identification models and classifiers is vast; however, an approach from scratch was chosen, along the implementation of a unsupervised model which helped to cluster no label data. Additionally, a reinforcement model capable to move bottles from different positions is planted for future development.**

As a result of the complexity of the project, two of three models got trained, concluding with two well dedicated models, and some ideas to continue enhance the project in the future.

***Index Terms*—Clustering, machine learning, bottle sorter, classification, computer vision, unsupervised model, reinforcement model.**

I. Introduction

Effective waste management and recycling has become important in the context of environmental sustainability and resource conservation. Among the various recyclable materials, plastics, especially PET (polyethylene terephthalate) and glass bottles make up a significant portion of the waste stream. Automating bottle identification, sorting and collection is critical to streamlining recycling processes and minimizing environmental impact.

To address this challenge, our project introduces an innovative approach that combines advanced computer vision techniques with the possible adaptive capabilities of reinforcement learning. Our main goal is to improve the efficiency of bottle sorting and collection in industrial and recycling environments. By leveraging computer vision technology, we achieve accurate bottle classification, distinguishing between PET bottles and glass bottles, laying the foundation for proper waste classification.

As we progress through this report, we will explore technical details, inherent challenges, and potential achievements of this bottle sorter classification project. From laying the theoretical foundations to the practical application of neural networks, this report will provide a comprehensive insight into how current technology is reshaping the way we interact with data and how these innovations can potentially create a lasting impact across various industries and fields of study.

II. Dataset characteristics

The final dataset consists of 983 images, where 501 are images of glass containers while the 482 remain for plastic container images. The original size of the images are 512x384 pixels; however, their size were reduced to 180x180 pixels to optimize the performance in the Clustering and CNN, and they are taken from various views and angles. The images are divided into two sets: a training set and a test set. The training set contains 786 images, and the test set contains 196 images. The training set is used to train the machine learning model, and the test set is used to evaluate the performance of the model.

In Figure 1, we can see a matrix of images giving a glimpse of the different views, types and forms of containers for each class; the process of segmentation and splitting of these data is explained in the data preprocessing section.

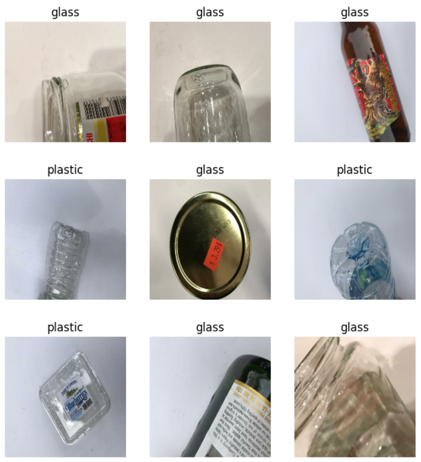

Fig. 1. Explicit plot of glass and plastic data, showing the relationship between images and labels

III. Data Preprocessing

The methodology followed to preprocess the information is divided into sections, where part of the coding is explained to understand what are their purpose of preparing the data for the type container detection model. This process involves loading image paths, segmentations, labeling, grouping, data splitting, and data analysis to ensure the data is ready for subsequent steps such as model training, and evaluation.

a) Data Handling:

The first thing is bringing the whole data information to be preprocessed, and analyzing the previous data labeled, by doing that we access the direction of the images in `data_dir`; in addition, the direction helps as a main path to find and create the training and validation sets to feed the model.

Using TensorFlow's functionality, it constructs training and validation datasets `train_ds` and `val_ds` from the provided directory. These datasets are segregated for training and validation purposes.

b) Data Optimization:

To enhance the efficiency of the data pipeline using TensorFlow's `tf.data` API. This optimization includes caching, shuffling, and prefetching data for both the training and validation datasets. Additionally, pixel values of the images within the dataset are normalized using TensorFlow's Rescaling layer to ensure consistency in model training.

IV. Modeling

A. Cluster unsupervised

A process of grouping or "clustering" of images is carried out using the K-Means algorithm. This method is used to classify image segments into different groups or clusters, with the aim of identifying patterns and similarities between them automatically. These images go through a preprocessing process, where they are converted to the Lab color space and the pixel values are scaled. Conversion to Lab helps separate luminance and chromaticity information, while scaling facilitates convergence of the clustering algorithm. In the context

of this process, image classification is carried out through the clustering technique known as K-Means. This approach seeks to identify patterns and similarities between image segments in an automated manner. The procedure starts with loading images from a specific location, followed by a preprocessing process. During this stage, each image is transformed to Lab color space and pixel values are normalized. This transformation favors the separation of luminance and chromaticity information, while normalization facilitates the convergence of the clustering algorithm. The SLIC (Simple Linear Iterative

Clustering) algorithm is then implemented to segment each image into coherent regions in terms of color similarity. This segmentation provides a partition of the image into segments, allowing focused analysis of specific areas rather than considering the image. Characteristics of each segment are extracted,

focusing mainly on the average color of said segments. These features are used as input to the K-Means algorithm, which groups the segments into a predetermined number of clusters. The assignment of cluster labels is done based on the similarity of the extracted features. Instead of a training and prediction approach, the K-Means algorithm assigns clusters to input data based on similarities in their features. In this context, "trainability" refers to fitting the K-Means model to the data through the clustering process, which seeks to minimize the intra-cluster variance and maximize the separation between clusters. The results of the clustering process are presented in the console, indicating whether each image is classified as a bottle or not, based on the cluster assignment. Additionally, the 'matplotlib' library is used to graphically display the original image, segmentation, and cluster assignment for each image, providing a visual representation of the results obtained through the clustering process.

The idea is to implement this solution in a band connected to one PC. This pc makes the decision depending on the result of the model. For that reason, the supervised model and unsupervised model was trained with the same dataset to identify if the bottle is a plastic or a glass bottle.

B. Convolutional Neural Networks (CNN)

Convolutional Neural Networks were developed to more effectively and efficiently process image data. This is due to the use of convolution operations to extract features from images. This is a key feature of convolutional layers, called *parameter sharing*, where the same weights are used to process different parts of the input image.

This allows us to detect feature patterns that are *translation invariant* as the kernel moves across the image. This approach improves the model efficiency by significantly reducing the total number of trainable parameters compared to fully connected layers.

The model architecture constructed within this script encompasses a Convolutional Neural Network (CNN) designed for image classification. Each layer within the network serves a specific purpose in feature extraction, abstraction, and classification.

1) Rescaling Layer

The Rescaling layer, the initial layer of the network, normalizes the pixel values of input images to a range between 0 and 1. This normalization standardizes the data, facilitating convergence during the model training process.

2) Convolutional Layers (Conv2D)

Following the Rescaling layer, multiple Conv2D layers are implemented. These layers perform convolution operations on the input images. Each Conv2D layer comprises learnable filters that scan the input images using a defined kernel size, extracting features such as edges, textures, and patterns. The activation function employed after each Conv2D layer is Rectified Linear Unit (ReLU), aiding in introducing non-linearity to the model's learned representations.

Conv2D Layer 1: Utilizes 16 filters with a 3x3 kernel size. It applies the ReLU activation function and maintains the spatial dimensions of the input through 'same' padding.

MaxPooling2D Layer 1: Immediately after each Conv2D layer, a MaxPooling2D layer is inserted to downsample the spatial dimensions of the feature maps, reducing computational complexity while preserving essential information. This layer uses a pool size of 2x2, reducing the resolution of the feature maps by selecting the maximum value within each pooling window.

Conv2D Layer 2: Comprises 32 filters with a 3x3 kernel size, applying ReLU activation and employing 'same' padding.

MaxPooling2D Layer 2: Similar to the previous layer, this MaxPooling2D layer downsample the feature maps derived from Conv2D Layer 2.

Conv2D Layer 3: Consists of 64 filters with a 3x3 kernel size, applying ReLU activation and maintaining spatial dimensions using 'same' padding.

MaxPooling2D Layer 3: Reduces the dimensionality of the feature maps produced by Conv2D Layer 3, aiding in abstraction and feature selection.

3) Flattening and Dense Layers

After the convolutional and pooling layers, the Flatten layer is introduced to reshape the 2D feature maps into a 1D vector, preparing the data for input into the densely connected layers.

Flatten Layer: Reshapes the high-dimensional feature maps into a 1D vector, preserving the learned spatial features while converting them into a format suitable for fully connected layers.

4) Dense Layers

The Dense layers are fully connected layers responsible for learning complex patterns and making final classification decisions.

Dense Layer 1: Consists of 128 neurons with the Rectified Linear Unit (ReLU) activation function. This layer learns higher-level abstractions from the flattened features extracted by the previous layers.

Dense Layer 2 (Output Layer): The final Dense layer contains the number of neurons equivalent to the number of classes in the classification task. It doesn't use an activation function, making it suitable for multiclass classification. The output of this layer represents the model's predictions for each class.[1]

C. Reinforcement Learning

In this section, it is discussed the approach that should be lead in order to achieve the reinforcement learning for the purpose planed.

1) Robot Setup

The first step is to select a robotic arm that is suitable for the environment application. Consider factors such as the mass of the entire system, materials being moved, and the reach of the robot arm 1.

Once, selected a robotic arm, it is necessary to calculate the link lengths of the manipulator. Using the forward kinematics to derive more insight into how the links are operating with respect to one another 2.

The next step is to generate a feasible path from point A to point B. This involves creating a set of connected waypoints that represent the path. it is suggested to use algorithms such as Point-to-Point Trajectories or Trajectory Planning for Robot Manipulators to generate a feasible path 34. Now, that it is generated a feasible path, is necessary to time scale the path to get a trajectory. This involves creating a time schedule for how to follow the path given constraints such as position, velocity, and acceleration.

2) Robot Training

For training the robotic arm, several types of reinforcement learning algorithms can be used. One of the most commonly used algorithms is the Deep Deterministic Policy Gradient (DDPG). This algorithm is an extension of the DPG algorithm and uses a deep neural network to approximate the value function and the policy. Another popular algorithm is Proximal Policy Optimization (PPO). This algorithm uses a neural network to approximate the policy and uses a gradient optimization method to update the policy. The Actor-Critic (AC) algorithm can also be used. This algorithm uses two neural networks, one to approximate the policy and one to approximate the value function.[3]

V. Results

A. Unsupervised cluster model

The result of the cluster model is a model trained in an automatic way to identify bottle passing over the band. The success of this cluster model lies in its ability to capture inherent patterns in the input data without explicit labels. By leveraging unsupervised learning through K-Means clustering, the model identifies commonalities in color, texture, and shape across the segmented images. Consequently, the model becomes a valuable tool for automating the identification process, showcasing its adaptability to variations in bottle appearance that may occur during production. With this model the solution is put a camera in the begin of the band using the model to interpret if in the waste there are bottle.

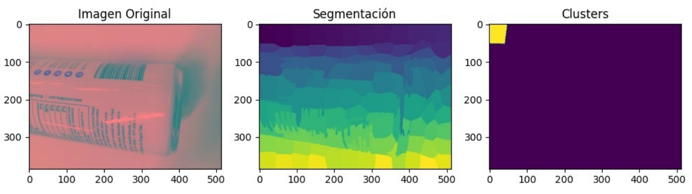

Fig. 2. bottle detection using cluster

B. Supervised CNN model

In the best experiment, we trained the model using the whole images for both training and validation. We utilized a batch size of thrity-two and ran fifteen epochs.

With the **CNN Sequential** architecture, we achieved a validation accuracy of 0.93, an accuracy 0.93 and a validation loss of 0.19 and a loss of 0.22; In this experiment, we can

conclude that the model is at least precise, and the results are quite similar. The model occasionally misidentifies images that do not contain a glass or a plastic container.

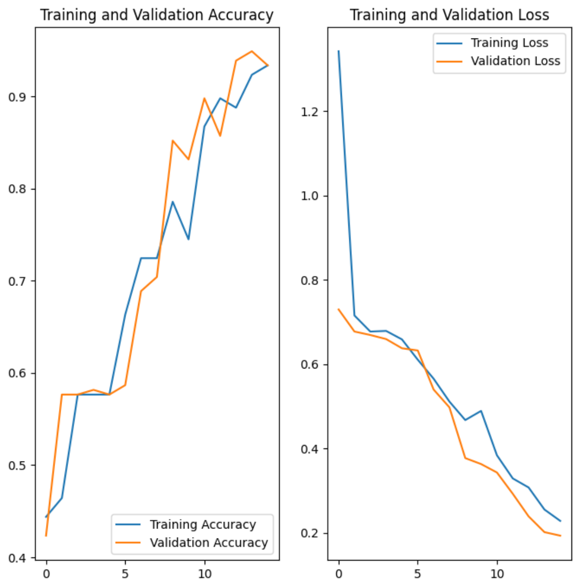

Fig. 3. Best results for accuracy and loss with the model

We created a model with the same images to create a cam visualization to classify if the container is made of plastic or glass. We use OpenCV to configure the camera to catch the images from it.

We saw that sometimes the model has confusion with PET bottles meanwhile the glass can detect all with a 90 percent confidence. Also, we want to try again this method using the model we created in TensorFlow, and try to create a dynamic classification.

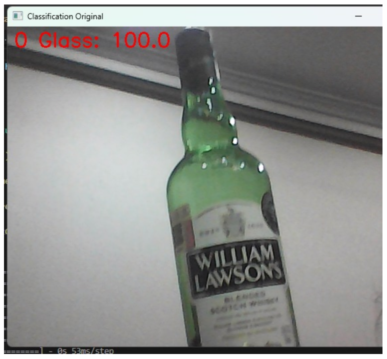

Fig. 4. Validation of Glass identification with its accuracy

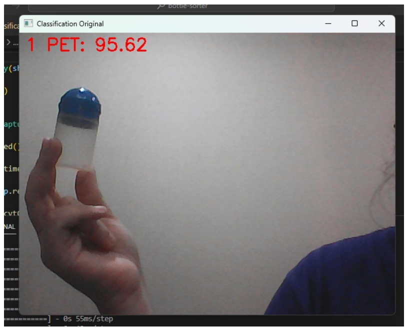

Fig. 5. Validation of PET identification with its accuracy

VI. Conclusion and Future Steps

To enhance our image classification model, we can consider strategies such as expanding the dataset, choosing more advanced neural network architectures, fine-tuning hyperparameters, utilizing data augmentation techniques, and continuously adapting the model to new challenges.

The next step would be to enhance its prediction capabilities and possibly apply it to work with real-time image processing.

As it was planned since the beginning, an implementation of a robotic arm would be fantastic for this project in real-world applications. The implementation of this ideas could be the next step that our model is needing to be fantastic. Nevertheless, to fulfill this, a reinforcement model is necessary, and the way it could be applied will be briefly described just below here.

To make a wonderful performance of the robotic arm, a pretty accurate trajectory needs to be satisfied for each bottle which is in optimal condition, and for this, the robotic arm needs to learn and reinforce about its own trajectories that it took in the past and make several comparisons of each one of them to reach a precise optimization of the better way that the arm should take to catch the bottle and save it for the next step.

VII. Appendix

Github Repository:

Bottle Sorter

https://github.com/Catunima/machine-sorter

References

[1] B. Kromydas, "Understanding Convolutional Neural Networks (CNNs): A complete guide," LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow with examples and tutorials, 18-Jan-2023. https://learnopencv.com/understanding-convolutional-neural-networks-cnn/. Accessed: Dec. 5, 2023.

[2] S. Dietrich. "Robot Motion Command Types: Understanding Linear, Joint, and Arc Movement - Technical Articles." Control.com - Forum for Automation & Control Professionals. https://control.com/technical-articles/robot-motion-command-typesexplained/. Accessed: Dec. 5, 2023.