Bottle Sorter Classification

Michelle Camara Gonzalez
2009014@upy.edu.mx
Eduardo Antonio Flores Arellano
2009055@upy.edu.mx
Julio Emanuel Gonzalez Gonzalez
1909075@upy.edu.mx
Ricardo Aron Tzuc Zih
1909177@upy.edu.mx
Computational Robotics Engineering
Universidad Politécnica de Yucatán
Km. 4.5. Carretera Mérida — Tetiz
Tablaje Catastral 7193. CP 97357
Ucú, Yucatán. México

Msc. Victor Alejandro Ortiz Santiago
Universidad Politécnica de Yucatán
Km. 4.5. Carretera Mérida — Tetiz
Tablaje Catastral 4448. CP 97357
Ucú, Yucatán. México
Email: victor.ortiz@upy.edu.mx

Abstract

In this project, we present a comprehensive solution to efficiently sort and collect PET and glass bottles in industrial and recycling environments. We face two fundamental challenges: bottle detection and classification using computer vision, along this report we discuss the process of solving those challenges, also we create the initials steps to deploy a computer vision system that uses cameras to detect and classify bottles into two main categories.

Index Terms

Engineering, programming, neural networks, AI, image classification, convolutional layers, convolutional neural networks, dataset, dataFrame, machine learning, supervised learning, bottle sorter

Bottle Sorter Classification

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 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 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.

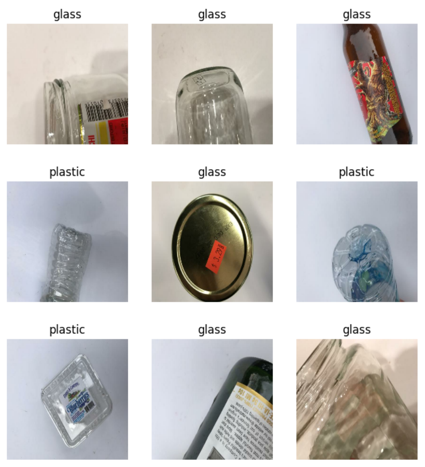

Fig. 1. Explicit plot of glass and plastic data, showing the relationship between images and labels

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

We will now describe the most important highlights of the model.

A. 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.

B. Model Development

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.

C. Training and Evaluation

The model is compiled using the Adam optimizer and the Sparse Categorical Crossentropy loss function for multiclass classification tasks. The training process involves iterating through epochs while evaluating the model's performance on the validation dataset. Metrics such as accuracy and loss are tracked for both training and validation sets.

V. Results

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.

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.

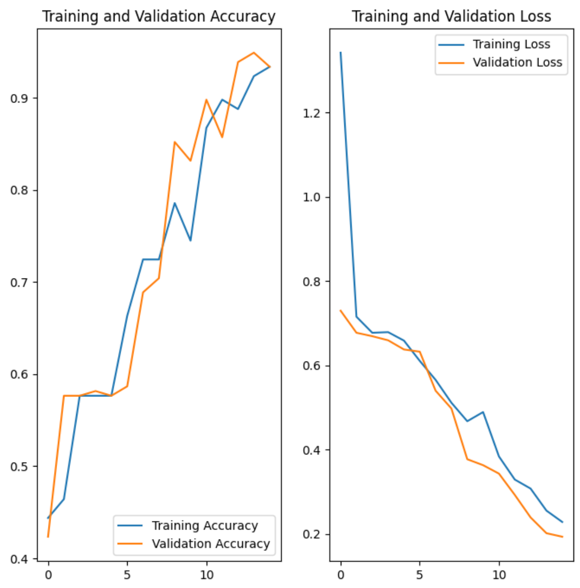

Fig. 2. Best results for accuracy and loss with the model

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 classificator.

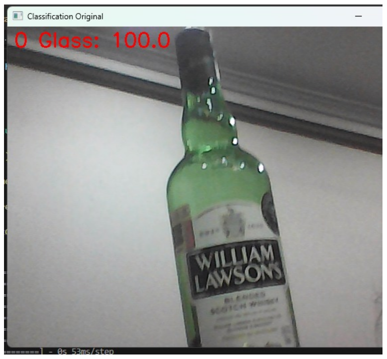

Fig. 3. Validation of Glass identification with its accuracy

VI. Conclusion and Next 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.

This system has the potential to operate in a real-world scenario due to its high accuracy. However, initially, it should function as an auxiliary tool. The next step would be to enhance its prediction capabilities and possibly apply it to work with real-time image processing.

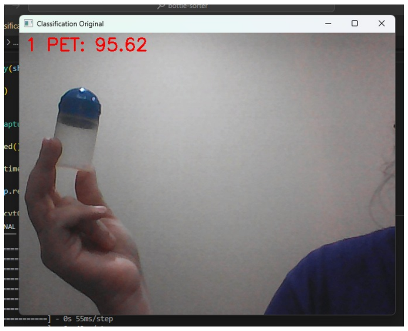

Fig. 4. Validation of PET identification with its accuracy

VII. Appendix

Github Repository:

Bottle Sorter

https://github.com/Catunima/bottle_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: 24-Aug-2023.

[2] Learnopencv.com. https://learnopencv.com/wp-content/uploads/2023/01/tensorflow-keras-cnn-vgg-architecture-1024x611.png. Accessed: 24-Aug-2023.