Spotting the Cap Using Deepnet

Ganesamanian Kolappan (9038581)

February 7, 2021

Abstract

Bottlecap detection is a beneficial task in the packaging industries. Bottle caps are popularly used across different fields ranging from mineral water manufacturers to the pharmaceutical industry. The process of localizing, detecting, and classifying the caps is an imperative task. This work's pivotal aspect is to detect and classify three classes of caps, namely, Faceup, Facedown, and Deformed. The proposed solution is robust to variations in the frame rates, illumination conditions, number of distractors, viewpoints, and background in the video input. This work aims to deploy a deep learning-based object detection methods that are MobileNet-SSD to detect caps. Besides, this work utilizes classical techniques for pre-processing, such as frame differencing for extraction of the stable frame and image enhancement techniques such as Contrast Limited Adaptive Histogram Equalization (CLAHE) from the video input. The caps are detected within a confined rectangular area already available in the stable frame extracted from video input. The trained MobileNet-SSD is used as a transfer learning for this work prepared again for this dataset. The developed Deepnet detector is evaluated using the metrics F1-score for the classification tasks. Deepnet detector localizes the caps with an efficiency of 80.1% and classifies them with a mean F-score of 0.89.

Index Terms

Deep learning, Single-shot object detector (SSD), MobileNet-SSD, F1-score, stable frame, frame differencing.

I. Introduction

Bottle caps are used to seal the bottles that prevent the oxygen from entering those bottles to avoid the material from oxidation [21]. Bottle caps are widely used in the packaging section in popular industries such as beer manufacturers, cool drinks manufacturers, and pharmaceutical industries [11]. Bottle caps are of different types, such as Crown cork, flip-top, screw cap, and plastisol [11]. In this work, the detecting screw cap is concentrated as it is widely used among the other types. Automatically detecting the caps provide information about the localization of the cap [21], proper sealing [11], and identify the defective cap [24]. This is highly beneficial for the packaging industry [12].

Localization and classification of objects in digital images are termed object detection in computer vision or image processing. There are numerous object detection methods available, ranging from traditional to deep learning [17][13]. Deep learning methods are successful in computer vision tasks. However, the choice of method solely depends on the approach and the nature of the problem [17][13]. This paper uses traditional methods for pre-processing and deep learning methods to localize and classify the screw caps into three classes: Facedown, Faceup, and deformed. Deep learning methods are advised for the extensive application where the data is enormous to train or in real-time application. The task handled in this work is simple that could be accomplished by the traditional method. Yet, certain complications could not be addressed using conventional methods like lighting and contrast variations and Region of Interest (RoI).

This work uses pre-processing such as frame extraction from the provided video(s), frame differencing to identify the stable frame, and an image enhancement technique called Contrast Limited Adaptive Histogram Equalization (CLAHE). These extracted stable frames (images) are learned by the deep learning method to detect the caps on the other new images. Advancement in deep architectures, especially Convolutional Neural Network (CNN) and the Graphical Processing Unit (GPU) for parallel computing, made the object detection handy. Deep learning methods are proven to be successful in object detection compared to the traditional methods that use SIFT [29] and HoG [16] for hand-engineered features. The capability of deep architectures based on CNN such as Inception [26], ResNet [6], VGG [25], DenseNet [8] to extract the maximum amount of features from the given image is the reason for the significant performance compared to the traditional method. These CNN architectures are the basic building block for most of the object detectors regarded as the state-of-the-art. Some of the object detectors are YOLO [18], YOLOv2 [19], and SSD [14], which have been successful with minor computations. Besides, object detectors, namely, Mask R-CNN [5], Faster RCNN [20], and R-FCN [3] based on regional CNN, have also proven successful.

Image processing using the deep learning method demands high computational hardware and larger memory. Model compression is a technique that stores the model efficiently with less memory consumption and requires comparatively less computational power. Methods such as MobileNets [7] and squeezeNet [9] are popularly used for model compression, which does not compensate for accuracy. MobileNet-SSD is relatively faster, requires less computation, less memory consumption, and competitive performance with other state-of-the-art methods [22]. Thus, this project work uses MobileNet-SSD [22] for spotting the bottle caps.

II. Problem Description

This work addresses classifying and localizing the three different classes of bottle caps, namely - Faceup, Facedown, and Deformed, in an image extracted from the video. The video includes the following steps:

1. Throwing any combinations of bottle caps along with the distractor such as coins, walnut, toys, and dice into a rectangular box (RoI).
2. Allowing the materials to settle down in a place without oscillations.
3. Picking up all the objects

The stable frame is extracted from this video, the frame where the caps are static without any movement/oscillations. This stable frame is the image where the MobileNet-SSD is used for classification and localization. The task is challenging due to the following reasons:

1. Each video in the dataset varies in shadow, frame rate, RoI, illumination, background, number of caps, viewpoint, and number of distractors. The distractors are various types, such as dice, clips, pin, nuts, coins, toys, and other objects. Figure 1 illustrates four difficulties among the above stated.
2. Inter-class variations where the object outside the box (RoI) is not considered. Additionally, in Figure 1 (b), the caps look like coins in the underexposed scenario. Whereas, in Figure 1 (a), coins reflecting light tends to be the cap facing upwards since the Faceup caps tend to be a reflective surface.

(a) Shadows and occlusion **(b)** Under exposed

Figure 1: *Difficulties in video data, the images shows the difficulties such as shadow, different view angle, reflective surface of coins and bottle caps, illumination effects.*

III. Related Work

Ufuk Sanver, Erdem Yavuz and Can Eyupoglu devised a approach [23] to detect the faulty bottle cap in the packaging of mineral water. The approach used sobel method and Hough transform for feature extraction as well as edge detection to detect the bottle cap from the other objects.

Zhang et al used Bayesian classifier [28] for detecting various types of bottle caps, the method was designed using OpenCV for image processing followed by the Bayesian classifier to identify the type of caps.

ZongFang Yang and JianYu Bai approached [27] as a two-step process: the image is smoothed to reduce the noise, then image is segmented using gray-level threshold and edge detection is applied.

Based on CNN a fruit recognition system is developed [2]. This work trains ResNet on the vegetable fruit dataset and the results are competitive to the state-of-the-art methods. However, the localization is undiscovered.

The research work [1] does a comparative analysis of various methods such as K-means clustering for edge detection and smoothing of image, classification models such as random forest, multilayer perceptron and support vector machines.

The existing image processing and learning based approach on bottle cap detection provides either classification or localization. Besides, there is no application that classifies three types of caps namely Faceup, Facedown, and deformed. This work has the potential benefit in the packaging sector and drinks or mineral water manufacturing industries. Therefore, the proposed work concentrates on the detection of Faceup, Facedown and deformed bottle caps in a video input. The detection is expected to be robust irrespective of the variations in frame rate, illumination conditions, viewpoints, and number of distractors of the video.

IV. Proposed Strategy

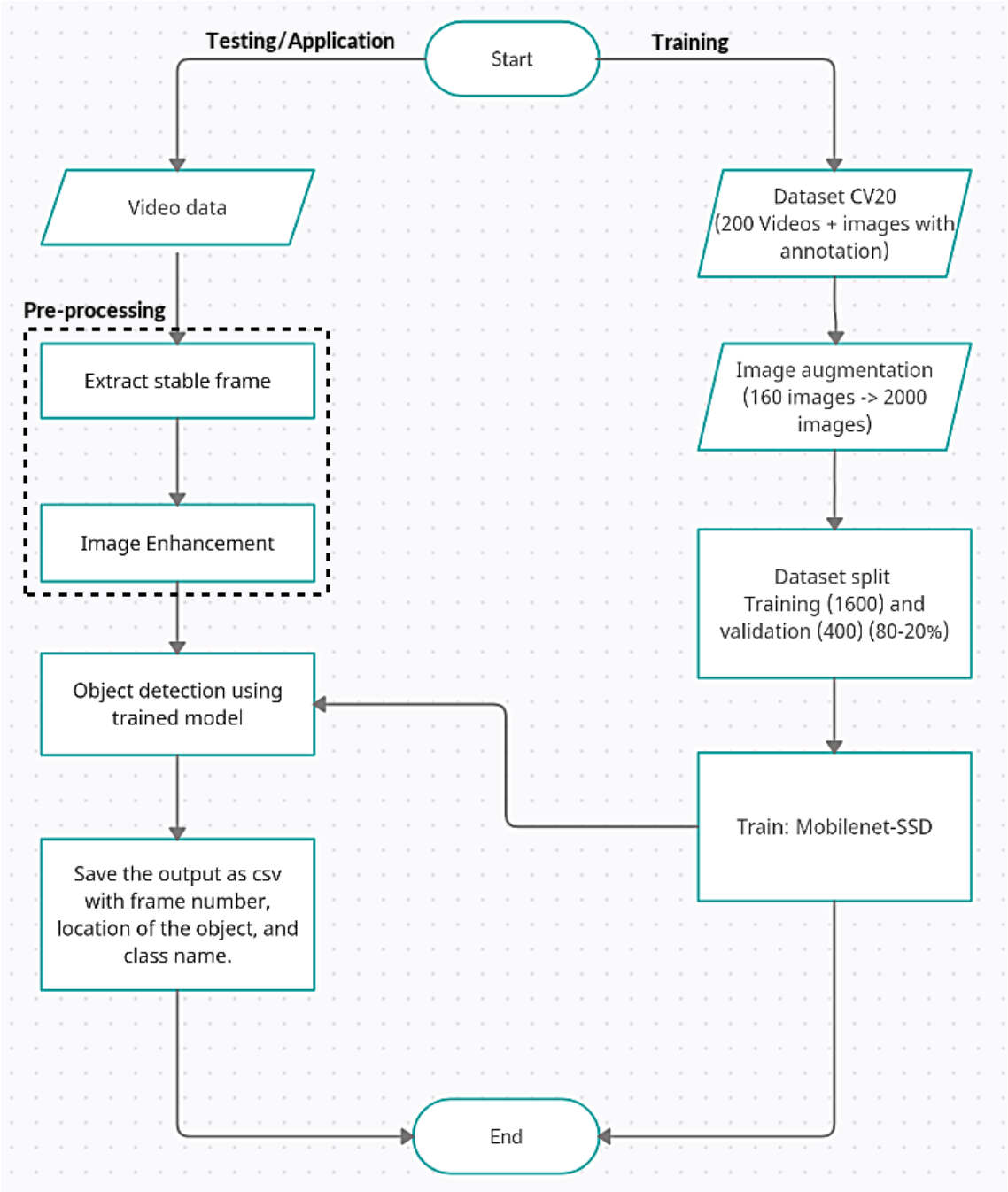


Figure 2: *Workflow of this work. Left side flow indicates the application stage where the trained model is utilized to detect the bottle caps. Right side flow indicates the training state where the model is trained to detect the bottle caps.*

The proposed strategy of this work is illustrated in Figure 2. The approach combines the conventional method for pre-processing and the deep learning method for object detection. The computer vision 2020 batch have collected a dataset consists of 200 videos and images with annotations. For training, images are sufficient to train the network, so the videos are not utilized. Since the method is a deep architecture, it requires a large amount of data. Image augmentation is the method used to increase the number of data to be trained. Among 200 images, randomly 160 images are selected, which are then augmented to produce 2000 images. The augmented images are split as 80-20% for training and validation, respectively. The model MobileNet-SSD is trained on the augmented images and saved for testing/application purposes. During testing, the video data is split into multiple frames. Among these frames, the stable frame is extracted using frame differencing using the OpenCV library. The extracted image is enhanced for better contrast for the trained model to detect the object. The saved model is executed on the extracted image after enhancement. The result is stored in the csv

file in the order of frame number, location (center x, y point of the object), and label sequentially.

V. Materials

To reproduce this work, a particular software package is necessary that has been used along with this work, such as Python >= 3.5 as the scripting language, OpenCV > 3.4.1 for pre-processing, Tensorflow = 1.15 model-MobilNet-SSD creation (already available model is used as transfer learning), LabelMe = 4.1.1 to annotate the image provided, Numpy >= 1.16 for mathematical works, Pandas >= 0.25 to store/read the data as an array, Scikit learn >= 0.22 for evaluation. Processing 2000 images require good computation power and memory capacity. The experiment is conducted on the Platform for Scientific Computing at Bonn-Rhein-Sieg University [4][10]. The server consists of hardware threads of 5,300 bridging 48,000 Graphical Processing Unit (GPU) cores with a collective memory of 460 TeraByte (TB) [10]. A subset of this configuration is made available for students' research work. The details are below in Table 1 [10].

Components	**Description**
Operating system	Nitrogen, Linux 7.8
Central Processing Unit (CPU)	50 nodes with 2x Xeon processors having 16 cores
Graphical Processing Unit (GPU)	Nvidia Tesla V100
Memory	50 GB for computation and 100 GB for storage

Table 1: *Machine configuration at Bonn-Rhein-Sieg University [4].*

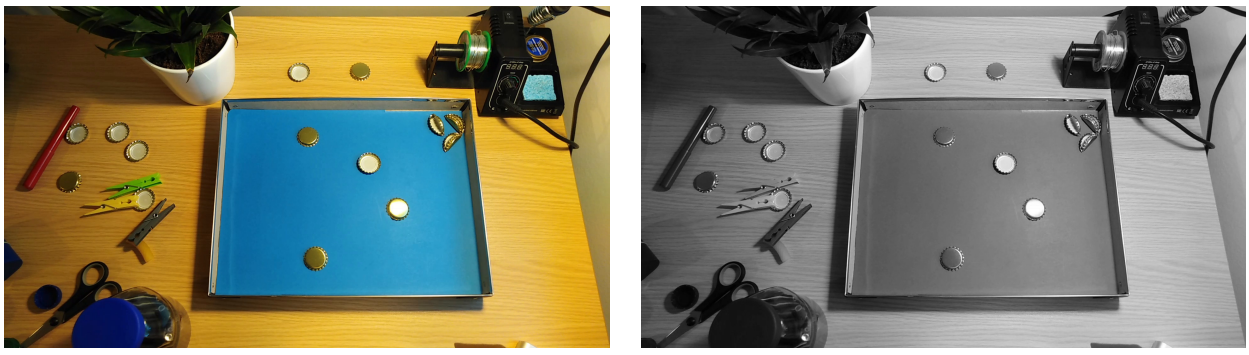
A good practice is to execute a small portion or subset of the code to check for error-free before performing on the University Computing Platform [10].

VI. Approach

This section details the approach used in this work.

1. Frame Extraction

The frame extraction is performed by the frame differencing followed by the Gaussian blur and canny edge. OpenCV is used to extract all the frames from the provided video data that varies from 114 to 463 frames depending on the video length and frame rate. These frames are stored as an array of images. The frame differencing is carried out between two successive images sequentially. The approach is devised so that there is a half second window for the static objects. The first and last few frames are ignored as the objects are dealt at the middle of the video. Besides, there is a chance of getting an empty box that can also be regarded as a static frame.



(a) Provided extracted image **(b)** Frame extracted by proposed approach

Figure 3: *Stable frame extraction the first image (a) is the reference image as provided. However, image (b) is extracted by the proposed method.*

The array of extracted images is converted into grayscale images using the cvtColor function as the computation cost and time is less to process. The successive grayscale images are paired sequentially and computed frame difference, then the image is processed using Gaussian blur of kernel size 3x3. A binary threshold is applied to the resultant image to remove the background shake, followed by the canny edge detection, which provides the proper suppression of background details from the object. As the below Figure 3 b) illustrates the

output of the above approach for the video "CV20_video_1.mp4". Whereas, Figure 3 a) is already provided for reference.

2. Image Enhancement

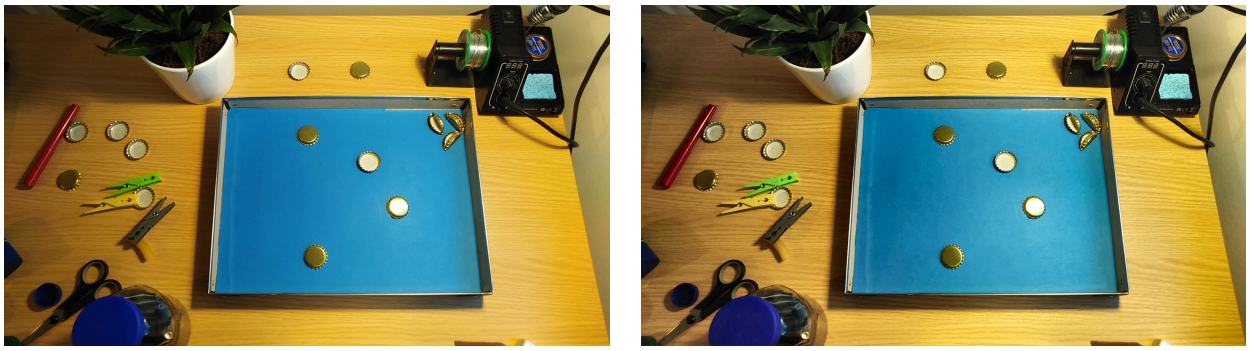
The datatset consists of few under exposed and negligible amount of over exposed videos. In order to overcome this issue, this work uses an image enhancement technique as follows:

1. Converting the RGM image to LAB color image.
2. Splitting the images into 3 channels for R, G, B respectively.
3. Applying CLAHE to the R-channel.
4. Merging the channels.
5. Converting back to RGB image.

This work have chosen an optimal parameter that works for both under exposed and over exposed videos.

(a) Provided extracted image **(b)** Image enhanced by the proposed approach

Figure 4: *Image enhancement for under exposed image, the first image (a) is the reference image as provided. However, image (b) is an underexposed which is extracted and enhanced by the proposed method.*



(a) Provided extracted image **(b)** Image enhanced by the proposed approach

Figure 5: *Image enhancement for slightly over exposed image, the first image (a) is the reference image as provided. However, image (b) is an slightly over exposed image extracted and enhanced by the proposed method.*

3. Data Augmentation

Data augmentation is done using TensorFlow functions. In many deep learning applications, data augmentation is momentous to teach the network to be robust across various input objects. To counteract, this work has generated more training data with patches of the original image with different crop ratios (e.g., 0.1, 0.3, 0.5, etc.) and random crop as illustrated in Figure 6. Besides, each image is also randomly flipped horizontally and vertically. Thereby, the objects' potential to appear in the right and left with similar likelihood has been addressed. The dataset consists of 200 images, from which 80% that is 160 images are chosen randomly for training. The selected 160 images are augmented to 2000 images and then split as 80-20% for training (1600) and validation (400), respectively.

Figure 6: *A single augmented image for Figure 5 b). The image is cropped with the factor 0.5 with horizontal and vertical flips.*

4. Training MobileNet-SSD

The augmented data is provided as input to the MobileNet-SSD [7]. The MobileNet-SSD is trained on the coco dataset shown in Figure 8. Since the model is already available and trained, this trained model can be used as transfer learning instead of building a model from the start. Thus, the SSD illustrated in Figure 7 has been used with the hyperparameters described in Table 2. Among the hyperparameters, tuning epochs, weight decay, learning rate, and batch size could improve the result, whereas changing other parameters will downgrade the results. The model is trained until the validation error becomes saturated to a minimum value (less than 0.1 as a metric loss).

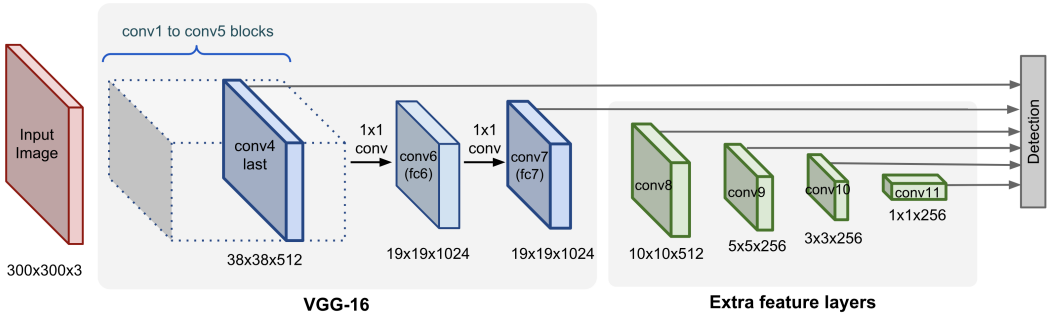


Figure 7: *SSD architecture [15].*

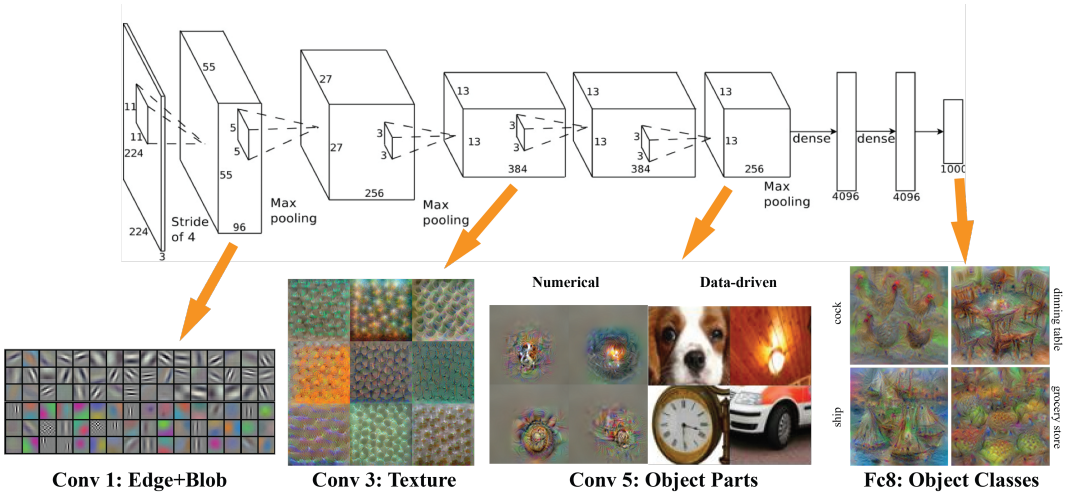
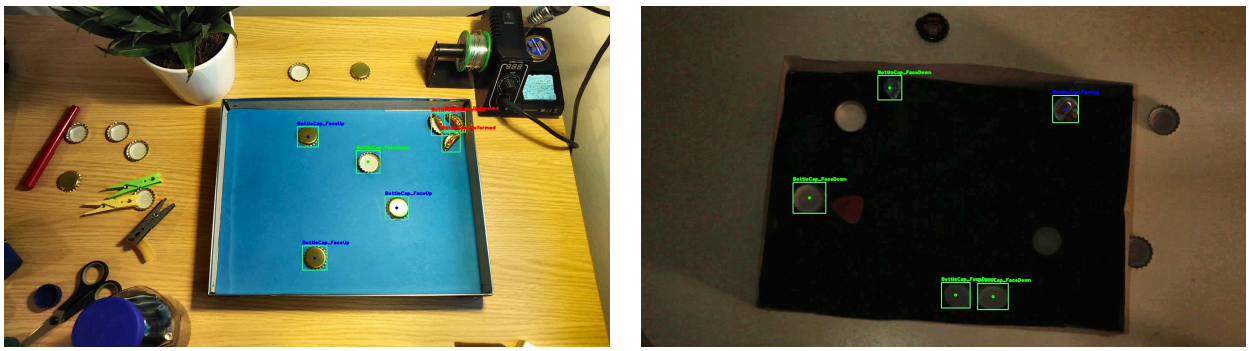


Figure 8: *Trained SSD feature map on the coco dataset [15].*

Hyperparameters	Value
Optimizer	Adam
Initial learning rate	0.001
Weight initialization	Xaviar
Batch size	32
Epochs	2000
Weight decay	0.0005
Learning rate decay factor	0.94

Table 2: *Hyperparameters for SSD.*



(a) Detection in slightly over exposed image **(b)** Detection in under exposed image

Figure 9: *Bottlecap detection using MobileNet-SSD.*

VII. Results and Discussion

The trained model is tested on the remaining 40 videos after the training. Before proceeding to the trained model's evaluation, the efficiency of the stable frame extraction is discussed.

1. Efficiency of Frame Extraction

Among 40 videos, the stable frame is extracted aptly for 36 videos. The failure cases are when the videos do have a frame rate of 30fps or when the whole action of throwing and picking up of the objects are dealt much faster with less settling time for the caps.

$$\text{Efficiency of Frame Extraction} = \frac{\text{Number of videos with correctly extracted frame}}{\text{Number of testing videos}} \text{X100} \tag{1}$$

Therefore, Efficiency of Frame Extraction $= \frac{36}{40} * 100 = 90\%$

(a) Objects are dealt fast **(b)** Higher frame rate - 30fps

Figure 10: *Failure in stable frame extraction.*

2. Evaluation of MobileNet-SSD

This research work is used for both localization and detection of bottle caps. The localization accuracy is given by the following equation

$$\text{Localization accuracy} = \frac{\text{Number of caps located correctly - Number of cap located incorrectly}}{\text{Total number of caps}} \text{X100} \quad (2)$$

The total number of caps in the 37 videos is 201, number of caps located correctly is 187, whereas number of caps classified wrongly is 26. Therefore,

$$\text{Localization accuracy} = \frac{187 - 26}{201} * 100 = \frac{161}{201} = 80.1\%$$

.

The classification of bottlecaps is evaluated by F1-score, which provides information about the enhancement in the performance of the method [10]. There are two main objectives; one is to minimize the incorrect classification (FP), which maximizes the precision, other is to minimize the incorrect missing(FN) to classify it correctly, which maximizes the recall. These two parameters provide the direction of enhancement.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

$$\text{F1-score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

Where, TP is True Positive, TN is True Negative, FN is False Negative, and FP is False Positive.

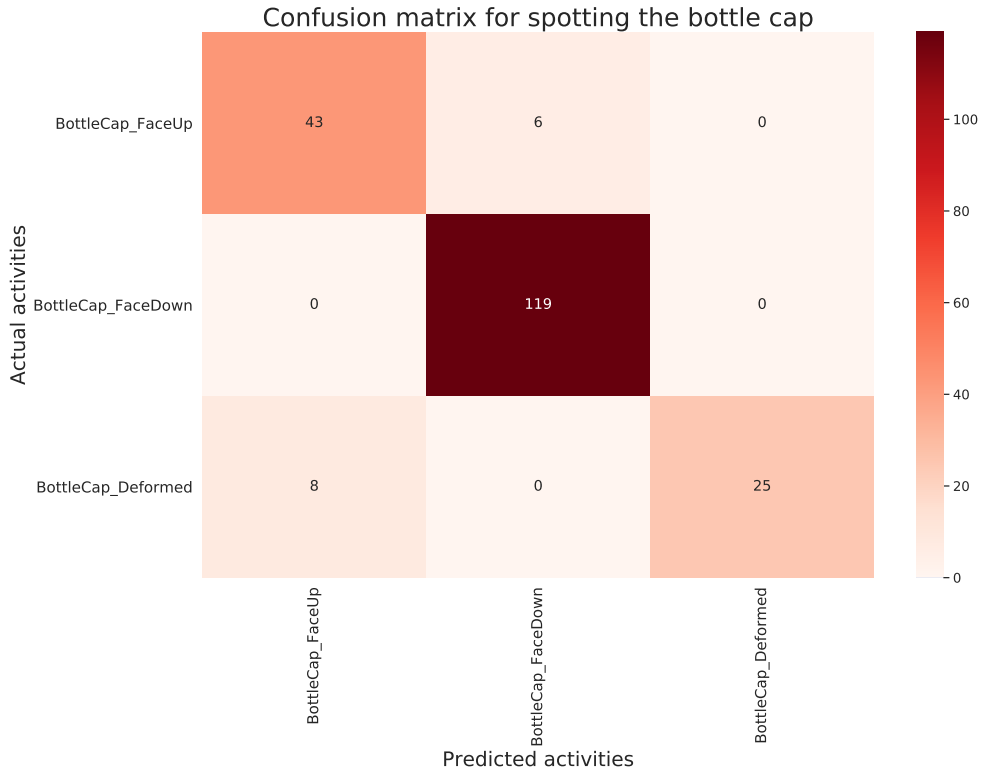


Figure 11: *Confusion matrix for spotting the cap [15].*

Figure 11 provides the confusion matrix to summarize the results. The F1-score value for each class is provided in the Table 3.

From the Table 3 it is evident that the performance on the classes Faceup and Deformed are to be improved. Whereas, the Facedown is detected satisfactorily.

Metric	Faceup	Facedown	Deformed
Precision	0.84	0.95	1
Recall	0.87	1	0.75
F1-score	0.85	0.97	0.85

Table 3: *F1-score for each classes.*

VIII. Future Work

Results of this works convey that the model performed insignificantly on the Deformed class of bottle caps. Following are the future works:

1. Data augmentation concentrated on a single object, such as a Deformed class object with different backgrounds and angles. Since the inbuilt Tensorflow method does the augmentation for the collective objects, the efficiency is low as there is confusion between Faceup and Deformed.

2. Robust frame extraction as the current frame extraction success rate is not 100% this may tend to fail in the particular application. Besides, this is the first step of any application. Therefore, it needs to be robust.

3. Should compare the performance of Yolo and classical method as these are competitive to the current work.

Bibliography

[1] M. Bahaghighat, F. Abedini, M. S'hoyan, and A. Molnar. Vision inspection of bottle caps in drink factories using convolutional neural networks. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 381–385, 2019.

[2] M. Buzzelli, F. Belotti, and R. Schettini. Recognition of edible vegetables and fruits for smart home appliances. pages 1–4, 09 2018.

[3] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *ArXiv*, abs/1605.06409, 2016.

[4] H. B.-R.-S. (H-BRS). Platform for Scientific Computing at Bonn-Rhein-Sieg University, 2020. Accessed on: 2020-12-21. [Online].

[5] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

[8] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[9] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *ArXiv*, abs/1602.07360, 2017.

[10] G. Kolappan. Feature extraction for motion data. Master's thesis, Hochschule Bonn-Rhein-Sieg, Germany, 2021.

[11] R. Kulkarni, S. Kulkarni, S. Dabhane, N. Lele, and R. S. Paswan. An automated computer vision based system for bottle cap fitting inspection. In *2019 Twelfth International Conference on Contemporary Computing (IC3)*, pages 1–5, 2019.

[12] W. Kumchoo and W. Chiracharit. Detection of loose cap and safety ring for pharmaceutical glass bottles. In *2018 International ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI-NCON)*, pages 125–130, 2018.

[13] A. Lee. Comparing deep neural networks and traditional vision algorithms in mobile robotics. 2016.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[15] X. Liu, Y. Tian, C. Yuan, F. Zhang, and Y. Guang. Opium poppy detection using deep learning. *Remote Sensing*, 10:1886, 11 2018.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[17] N. O. Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernández, L. Krpalkova, D. Riordan, and J. Walsh. Deep learning vs. traditional computer vision. *arXiv: Computer Vision and Pattern Recognition*, 2019.

[18] J. Redmon, S. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[19] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[20] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

[21] X. Ren, J. Wen, Y. Lan, T. Li, and X. Wang. Design of bottle cap detection system based on image processing. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 4880–4885, 2020.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[23] U. Sanver, E. Yavuz, and C. Eyupoglu. An image processing application to detect faulty bottle packaging. In *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 986–989, 2017.

[24] Y. Shen, R. Mo, L. Wei, Y. Zhu, Z. Peng, and Y. Zhang. Bottle cap scratches detection with computer vision techniques. In *2013 Ninth International Conference on Natural Computation (ICNC)*, pages 1314–1318, 2013.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[27] Z. Yang and J. Bai. Vial bottle mouth defect detection based on machine vision. In *2015 IEEE International Conference on Information and Automation*, pages 2638–2642, 2015.

[28] R. Zhang, Y. He, P. Zhang, Y. Hu, Y. Cao, and J. Zhang. The bottle size detection system based on bayesian classifier. In *2015 IEEE International Conference on Information and Automation*, pages 1157–1162, 2015.

[29] Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. *ArXiv*, abs/1905.05055, 2019.