Logo Detection and Recognition

Viet Nam National University, Ho Chi Minh City
University of Science
Faculty of Information Technology
Computer Vision – 21TGMT

Group 05 Members:
Vo Nguyen Hoang Kim - 21127090
Tran Thanh Ngan - 21127115
Lam Thanh Ngoc - 21127118

Instructors:
Vo Hoai Viet
Pham Minh Hoang
Nguyen Trong Viet

***Abstract*—Logo detection and recognition have become pivotal tasks in various applications, serving to establish brand identity and combat unauthorized logo usage. Despite significant contributions from numerous authors, traditional methods reliant on edge and shape detectors encounter challenges in accurately detecting and recognizing logos due to variations in shape, size, location, and occlusion issues across different images. In this paper, we present a comprehensive literature survey covering both traditional and recent advancements in logo detection and recognition techniques. Specifically, we explore the utilization of different descriptors such as SURF, SIFT, HOG, and the hybrid descriptor ORB, even Deep Learning approaches, aiming to surpass the limitations of conventional approaches. This review aims to provide researchers engaged in logo detection and recognition, as well as object recognition, with a comprehensive understanding of the field, offering insights into the latest methodologies and developments.**

Keywords—logo detection, logo recognition, descriptor, Deep Learning, Convolutional Neural Network, YOLO.

I. INTRODUCTION

In the digital era, logos have emerged as quintessential elements of visual communication, serve as powerful symbols that encapsulate identity and values from global brands to local businesses. Logos typically consist of texts, shapes, images, or combinations thereof. Logo detection and recognition in images and videos has been gaining considerable attention in recent years due to their applications in copyright infringement detection, intelligent traffic-control systems, and automated computation of brand-related statistics.

The primary task of logo detection involves locating specific logos in images or videos and identifying them. Traditional methods often utilize handcrafted features (like SIFT) and statistical classifiers, which are suitable for well-defined shapes and affine transformations, like those found in the domain of logos. However, these methods suffer from complex image preprocessing pipelines and poor robustness when dealing with a large number of logos.

In real-world scenarios, logo detection faces challenges since numerous brands may have highly diverse contexts, varied scales, changes in illumination, size, resolution, and even nonrigid deformation. To address these challenges, recent research has investigated Deep Learning, which has gained success since ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Deep learning-based solutions with expressive feature representation capability offer better robustness, accuracy, and speed and thus attract increasing attention.

This survey aims to provide a comprehensive overview of the state-of-the-art techniques, methodologies, datasets, and applications in logo detection and recognition.

The rest of this report is organized as follows. Literature survey is presented in sections II and III; while Section II covers some traditional methods, Section III investigates deep learning approaches. In Section IV, we review the public logo detection datasets, then propose a method in Section V. Finally, we provide our experiment, analyze and evaluate in Section VI, and lastly, conclusion in Section VII.

Figure 1. Example of global brands and local businesses logo

II. DETECTION & RECOGNITION USING DESCRIPTORS

After successfully detecting the presence of logos within images, the subsequent step involves their recognition. While there exist many techniques for logo recognition, our approach in this report centers around the extraction of distinct features from the input images. These features are then compared against a repository of features extracted from logo database. Through this comparative analysis, we aim to identify detected logos. This methodology is described in *Figure 2*, showcasing the general process of logo recognition.

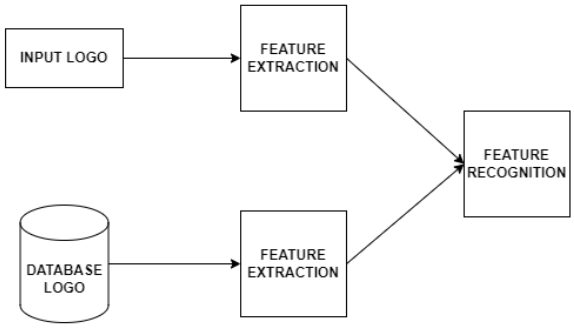

Figure 2. General Logo Recognition Framework

A. *Descriptor*

Below is a brief analysis of descriptors utilized in logo recognition or matching research papers, which are pivotal for extracting distinctive features from logos, enhancing recognition accuracy across various applications. Through this analysis, we aim to delineate their effectiveness and limitations in logo recognition and matching tasks.

a) SIFT descriptor: SIFT (Scale Invariant Feature Transform) is used to detect and describe local features in the image. [6] SIFT detects a large number of features from images, which reduces the errors caused by local variation in the average error of all feature matching errors.

b) SURF descriptor: SURF (Speeded Up Robust Feature) [7] is a fast descriptor as compared to SIFT. SURF is fast because it extracts features per region, whereas SIFT extracts features of whole image so it takes more time, but SURF extracts a smaller number of features as compared to SIFT. There are some reports that show the comparison between SURF and SIFT. SURF is fast compared to SIFT but it detects less key points as compared to SIFT. SURF is fast because it uses integral images for fast calculation.

c) HoG descriptor: *HoG (Histogram of Oriented Gradients) analyzes the distribution of edge orientations within an object to describe its shape and appearance. The HOG method involves computing the gradient magnitude and orientation for each pixel in an image and then dividing the image into small cells.*

d) Hybrid descriptor: *Hybrid descriptor is the fusion of two or more than descriptors. In hybrid version feature extracted from different descriptors are combined for fast matching. For instance, a method proposed by Chinmoy Biswas (2015) [3] combine extracted features of SIFT, SURF and HOG for logo recognition.*

B. Matching

Matching in object recognition is the process of comparing the features extracted from an object of interest with those of known objects to identify similarities or matches. After feature extraction, which involves capturing the distinctive characteristics of the object, the matching task aims to find the best correspondence between these features and those stored in a reference database or learned during training. Below are some types of Matching that we have documented.

To facilitate matching, many methods have been proposed and used in the field. These encompass diverse approaches such as Consensus-based Matching Score (CDS), Random Sample Consensus (RANSAC), and distance metrics including Euclidean and Manhattan distance. Each method offers unique advantages and trade-offs, catering to different scenarios and requirements in the matching process. Through the utilization of these methodologies, researchers aim to achieve robust and accurate matching results across various applications in computer vision and recognition.

C. Comparision

The following table was referenced from [12] and documented using statistical data derived from various research papers. It offers a comprehensive overview of the methodologies utilized in each experiment, providing insight into the specific approaches adopted by researchers. Furthermore, it presents the evaluation results provided by the authors, offering valuable insights into the effectiveness and performance of the methods employed in different experimental settings.

RELATED WOKS

Name of paper	**Method used**	**Claim by author/ remark**
Logo Recognition Technique using Sift Descriptor, Surf Descriptor and Hog Descriptor (2015) [3]	Extraction: SIFT, SURF, HoG Matching: Manhattan distance	High accuracy, but illumination problem
Logo Matching And Recognition System Using Surf (2014) [9]	Extraction: SURF Matching: CDS	Focus on accuracy, complexity and time
Context-Dependent Logo Matching and Recognition (2013) [6]	Extraction: SIFT Matching: CDS	Focus validity and key-point matching

III. DETECTION & RECOGNITION USING DEEP LEARNING

Some traditional methods are applied such as: SIFT, HOG or SVM. However, these methods are not effective in detecting logos in the real world which can not only cause low relevance with high time complexity but also be inappropriate for the diversity data. This is also the reason for the research of deep learning in logo detection. Due to the appreciation for reality of various contexts, there are many different detection algorithms and the two may be used for a large amount of context are introduced below.

Multi-scale Logo Detection

The Multi-scale Logo Detection is supposed one of the modern algorithms that goes along with the **Feature Pyramid Networks (FPN)** introduced in 2017 to solve the problem of scale invariance with higher performance quality of small-scale logo and keeping the amount of calculation.

Discussing about the architecture, it is a top-down architecture with lateral connections that is used to build high-level semantic feature maps at all scales from a single resolution input image and focuses essentially on the problem of scale invariance and the loss of information caused by the convolutional neural network (CNN) or region proposal network (RPN).

Figure 3. Simplified logo classification pipeline

The top-down architecture can be described with the 2 constructions: the bottom-up pathway and the top-down pathway. While the bottom-up pathway is the same as the standard convolutional network that is used to build a feature pyramid from a single resolution input image, the top-down pathway is used to build high-level semantic feature maps at all scales from the coarsest level to the finest level.

The bottom-up pathway starts with applying a series of convolutions and pooling operations to reduce the resolution of the feature maps and increase the number of channels by "going up" the pyramid. Its usage is to extract the feature maps at different scales.

The top-down pathway is used to build high-level semantic feature maps at all scales from the coarsest level to the finest level. It is done by applying a series of convolutions and upsampling

operations to increase the resolution of the feature maps and decrease the number of channels by "going down" the pyramid. The lateral connections are used to combine the feature maps from the bottom-up pathway with the feature maps from the top-down pathway to build the final feature pyramid.

There are many algorithms that have been done to improve the performance of the Multi-scale Logo Detection such as the **Faster R-CNN**, the **YOLO**, the **SSD**, etc. The Faster R-CNN is a region-based convolutional neural network that is used to detect the logo in the image. It is composed of 2 modules: the region proposal network (RPN) and the Fast R-CNN. The YOLO (You Only Look Once) and the SSD (Single Shot MultiBox Detector) are real-time object detection systems that are used to detect the logo in the image. They are used to predict the bounding boxes and the class probabilities directly from the full image in one evaluation. Based on the combination of these works with the Multi-scale Logo Detection, a number of potential works have been implemented and proposed such as the **obtaining sufficient features for logo (OSF-Logo)** with the **Regulated Deformable Convolution (RDC)** module, the **multi-scale feature decoupling network (MFDNet)**, **multi-scale vehicle logo detection (SVLD)**, etc.

Logo Detection in Complex Scenes

One of the highlighted applications of logo detection is identifying it in such complex scenes including movements, natural conditions, perspectives, etc. Yuan Liao et al. [1] has introduced the mutual enhancement for detecting of multiple logos in sports video by analyzing the homogeneous logos occurrences and the motion-estimation-based propagation for the qualities of logos may be affected by many factors which are the distance to the camera or the perspective distortion.

Although there are various methods recommended for such the challenge and shows prospective experiments, the problem of logo detection in complex scenes is still being considered as the real-world data is more and more diverse and complex which is supposed to be not ideal. For that reason, the LogoNet framework is proposed as an effective solution.

From Jain et al. [2], the LogoNet is inspired by the HourGlass Network for extracting backbone with the different output map of features and the additional module called spatial-attention for accuracy. The detail of the LogoNet framework implementation is described by Jain et al. as the figure 4:

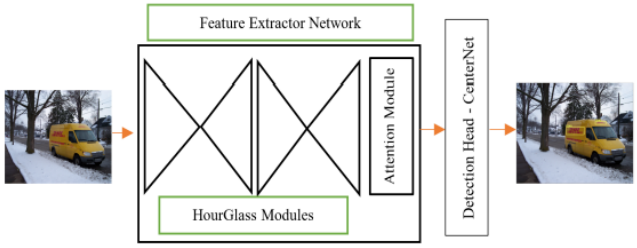

Figure 4. General architecture of the LogoNet framework

IV. DATASET

There are many datasets used to evaluate the performance of detection and recognition algorithms. In this report, we propose the following suggestions:

1) ***MICC-Logos:*** This dataset contains 720 images downloaded from the web; it contains 13 logo classes each of them represented with 15-87 real world pictures.
2) ***Flickr Logos:*** This dataset contains logos of 32 different logo brands downloaded from Flickr.
3) ***Tobacco-800 Signatures and Logos:*** This dataset is composed of 1290 document images. Dimensions of images are from 1200 by 1600 to 2500 by 3200 pixels.
4) ***Roboflow Logo-7:*** This Dataset comprises 4,129 images spanning across 53 classes, including prominent brands like Adidas, Chanel, and other.

V. RECOMMENDATION

Upon survey, we've found that utilizing multiple models for each task is quite costly and time-consuming. With the remarkable development of neural network models, our team considers leveraging them to simultaneously perform both tasks. We've observed that YOLO family models excel at this. Therefore, implementing YOLO models sounds approach for both detection and recognition tasks on logo datasets.

VI. EXPERIMENT

A. YOLOv5 model

1. Overview

YOLOv5 is the fifth iteration of the revolutionary "You Only Look Once" object detection model, is designed to deliver high-speed, high-accuracy results in real-time. It comes in four main versions: small (s), medium (m), large (l), and extra-large (x), each offering progressively higher accuracy rates. Each variant also takes a different amount of time to train.

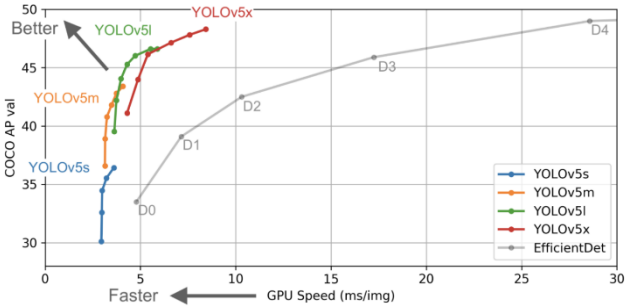

Figure 5. Experimental results with YOLOv5 (according to YOLOv5 - README in this repo)

In the above chart, the goal is to produce an object detector model that is very performant (Y-axis) relative to its inference time (X-axis). Preliminary results show that YOLOv5 does exceedingly well to this end relative to other state of the art techniques.

YOLOv5 derives most of its performance improvement from PyTorch training procedures, while the model architecture remains close to YOLOv4.

2. Architecture

YOLOv5 is different from the previous releases in that YOLOv5 utilizes PyTorch instead of Darknet. It utilizes CSPDarknet53 as backbone. It uses Path aggregation network (PANet) as neck to boost the information flow. PANet adopts a new feature pyramid network (FPN) that includes several bottom ups and top-down layers. This improves the propagation of low-level features in the model. PANet improves the localization in lower layers, which enhances the localization accuracy of the object.

In addition, the head in YOLOv5 is the same as YOLOv4 and YOLOv3 which generates three different output of feature maps to achieve multi scale prediction. YOLOv5 model can be summarized as follows: Backbone: Focus structure, CSP network, Neck: SPP block, PANet, Head: YOLOv3 head using GIoU-loss. The improvement of YOLOv5 over YOLOv4 was utilization of CSPDarknet53 which solved the problem of repetitive gradient information found in YOLOv4 and YOLOv3 thereby reducing the network parameters and reducing inference speed while accuracy is increased. The architecture of YOLOv5 is shown in Figure 6.

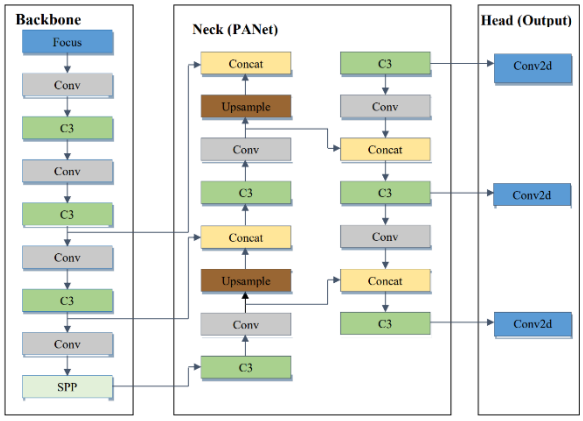

Figure 6.YOLOv5 architecture

B. Dataset

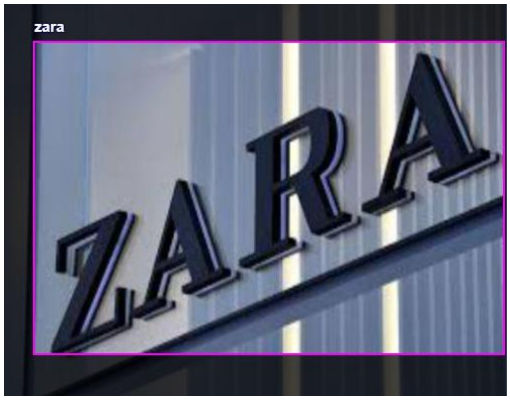

Figure 7. Example of an annotated image in roboflow

Logo Detection Dataset (from roboflow) comprises 4,129 images spanning across 53 classes, including prominent brands like Adidas, Chanel, Calvin Klein, Uniqlo, among others. Notably, this dataset is meticulously annotated, ensuring each image is accurately labeled with its corresponding class. Moreover, to facilitate model training and evaluation, the dataset is efficiently splitted into distinct subsets: the training set, consisting of 82% of the data (5,782 images), the validation set, comprising 12% (825 images), and finally, the test set, which encompasses the remaining 6% (413 images). Total number of annotations in the dataset is 5,002. On average, each image has about 1.2 annotations. *(as shown in Figure 8, 9).*

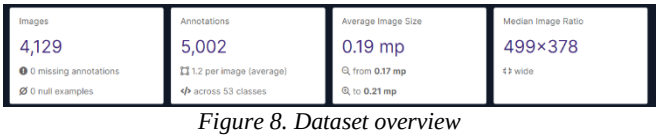

Figure 8. Dataset overview

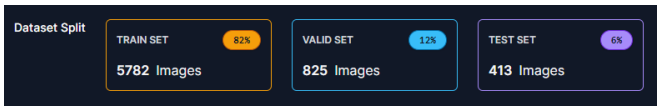

Figure 9. Dataset split

However, upon closer examination, Figure 10 describes the uneven distribution of classes within the dataset. Some classes exhibit an overrepresentation, indicating a surplus of instances relative to others, while certain classes are underrepresented, suggesting a scarcity of instances.

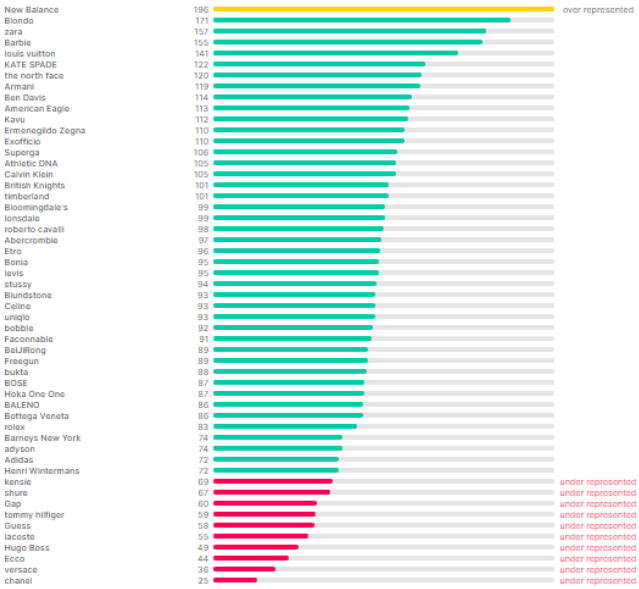

Figure 10. Dataset exploration

For example, the class “New Balance” means that it is overrepresented in the dataset with 196 samples. This suggests that there is an excessive number of images or data samples belonging to the "New Balance" class compared to other classes. Similarly, the class “Chanel” signifies that it is underrepresented in the dataset. With only 25 samples, this class has fewer representations compared to other classes. Therefore, both result in an imbalance in the dataset, indicate potential challenges in training models and evaluating performance.

Overall, this dataset appears to be comprehensive, well-annotated, and suitable for various computer vision tasks, specifically for logo detection and recognition.

C. Training

With the logos dataset (described and annotated above), we used the YOLOv5 model to conduct training for logo detection and recognition. To execute the model training, the command structure is invoked as follows:

```
!python train.py --img <image_size> --epochs <no_epochs> --data <path_to_dataset> --weights <pre-trained_weights>
```

where:

- --img: Represents the size of the image, here we are using 640.
- --epochs: The number of epochs for model training, here we are using 100.

- data: The path to the dataset, here is the file data.yalm (this file can be found within the dataset).
- --weights: The pretrained weights are auto downloaded from the latest YOLOv5 release. Here we are using yolov5s.pt (recommended according to Ultralytics).

However, due to resource limitations in the Google Colab environment (especially with the free GPU instance), training encounters difficulties when attempting to execute up to 100 epochs. Despite numerous attempts, the model typically reaches only around 54 epochs during training for this dataset. During execution, the model records two sets of weights: "last" and "best". "last" represents the weights at the end of the final epoch of training, while "best" represents the best weights encountered during training. These files typically have the extension .pt *(last.pt* and *best.pt).* The figure below illustrates the training process at each epoch. It provides essential metrics for evaluation and facilitates easy monitoring of the model's learning progress.

```
Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
19/99     4.92G    0.04283    0.01631    0.02358         17        640: 100% 362/362 [03:34<00:00,  1.69it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 26/26 [00:13<00:00,  1.91it/s]
             all        825       1002      0.829      0.752      0.807      0.434

Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
20/99     4.92G    0.04219    0.01601    0.02275         23        640: 100% 362/362 [03:33<00:00,  1.70it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 26/26 [00:12<00:00,  2.10it/s]
             all        825       1002      0.825       0.74      0.801      0.464
```

Figure 11. Training process at each epoch

At each epoch, it provides values for loss metrics (such as Box loss, Object loss, Class loss) as well as accuracy metrics (mAP50, mAP50-95). This helps us easily assess the correctness and rationality of the model during the learning and training process. We can also observe these metrics over the entire training process, which is done and visualized using TensorBoard. These figures 12 below are the charts visualized based on the metrics obtained throughout the entire training process (specific numbers can also be viewed through the attached csv file in **Document** folder).

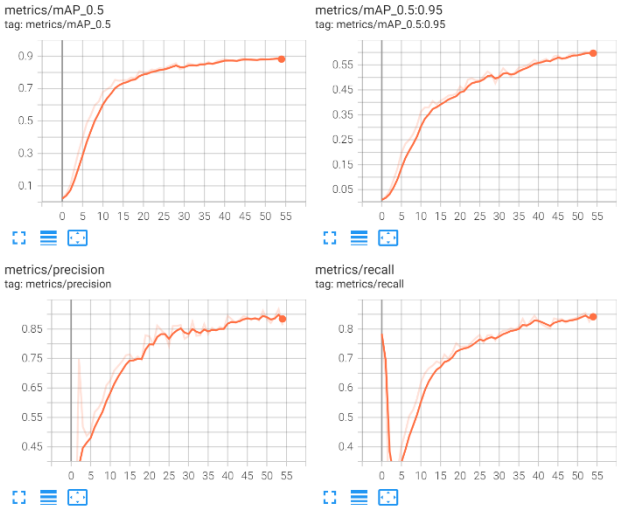

Figure 12-1. Metrics

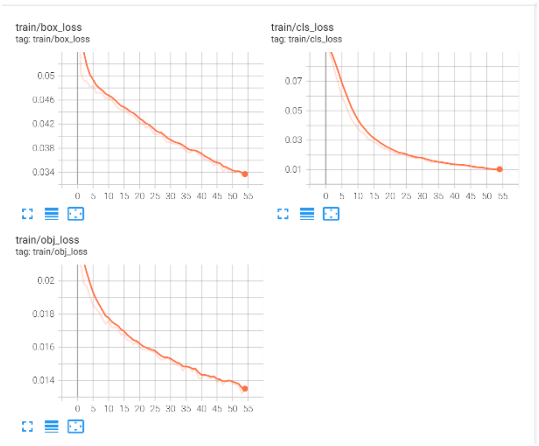

Figure 12-2. Train/Loss

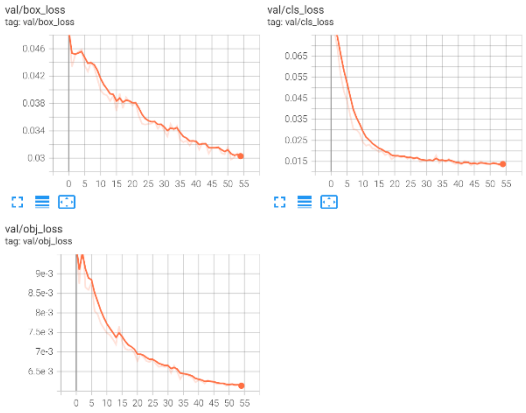

Figure 12-3. Validation/Loss

Assessment: Regarding the metrics, the AP, mAP (at 0.50 and 0.50-0.95), and Recall values demonstrate steady improvement with each epoch, occasionally experiencing minor fluctuations that are not substantial. Moreover, the model has excelled in minimizing errors not only during training but also during validation. Despite training for a modest number of epochs (approximately 54), overview, the model has achieved commendable performance levels in both logo detection and prediction.

Improvement: After a week of attempting to develop a model to achieve the desired performance without facing GPU resource issues on Google Colab, we found that reusing weights (best.pt file) trained from the previous session to continue training showed promising results, rather than having to train from scratch with increased epochs. Below are the results we achieved when training the model with 50 epochs using the weight obtained from the previous training session (best.pt file) instead of using the pre-existing weights of the YOLO model (yolov5s.pt file).

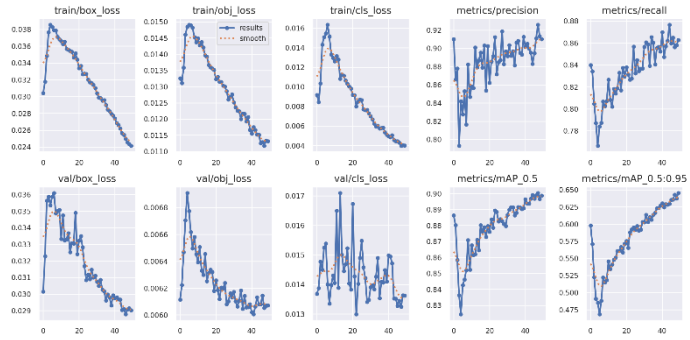

Figure 12-4. Training process diagram with best.pt

Loss: Despite a slight increase in loss during the initial epochs (approximately the first 5 epochs), it gradually decreases thereafter, optimizing through each epoch, resulting in the model nearly converging with very minimal loss.

Metrics: In terms of metrics, including AP, mAP (at 0.50 and 0.50:0.95) and Recall, it can be observed that the values are somewhat higher compared to the previous training session (training for 55 epochs with weights as yolov5s.pt), although the difference is not significant. Nonetheless, it still yields quite satisfactory results.

Therefore, we can utilize this technique to conduct model training to conserve GPU resources while simultaneously enhancing the model's performance during training.

D. Prediction

The detection of objects in images is done using the best trained weights of the YOLOv5 model, which is saved as the **`runs/train/exp/weights/best.pt`** file. The valid command is used to detect images with the size of 640x640 pixels and confidence threshold of 0.4. Images that are used for source are taken from the `test` folder of the dataset presented above. The detected objects are shown with bounding boxes and labels in the images saved in the `runs/detect/exp` directory.

As can be seen from the results, the YOLOv5 model is able to detect almost all the objects in the images with high accuracy that up to nearly 90%. Compared to the other object detection models, YOLOv5 can detect exactly the logo without costing much time or resources as CPU or GPU.

Figure 14. Detected objects images

The results also explain the difference of the accuracy weights that depend much on the perspectives and the most ideal view that gives to the best weight is the front-facing. In general, brands are detected from almost all points of view.

F. User guide

The experiment is implemented on Ubuntu with NVIDIA (R) Cuda compiler driver. The CUDA device used is NVIDIA GeForce MX330.

The detail guides for using YOLOv5 to detect objects as well as getting the logo dataset from **'roboflow'** is shown in the .ipynb file. The executable file is attached with the link that can be called by terminal with the valid command as below:

wine {path_to_exe}/detect --weights {path_to_weight}/best.pt --img 640 --conf 0.4 --source {path_to_test_image}

In which:

- wine: a compatibility layer that allows you to run windows applications on Linux. In this case, we are using wine to run the yolov5 model.
- {path_to_exe}/detect: the path to the detect.py file in the yolov5 repository.
- --weights {path_to_weight}/best.pt: the path to the weights file of the trained model. We are using the best weights file from the training results.
- --img 640: the size of the input image. In this case, we are using an input image size of 640x640 pixels.
- --conf 0.4: the confidence threshold for detection. Only detections with a confidence score greater than 0.4 will be shown.
- --source {path_to_test_image}: the path to the input image that we want to detect the logo in.

After being improved, the statistics are shown in the 4 csv files which are 2 for prediction and 2 for the results of the training process.

VII. CONCLUSION

In this work, our experiment on logo detection using YOLOv5 has demonstrated its effectiveness and suitability for practical applications. Its combination of accuracy, speed, and robustness makes it a compelling choice for logo detection tasks across various industries. Additionally, YOLOv5 displayed resilience to variations in logo size, orientation, and background clutter, underscoring its robustness and adaptability to real-world scenarios. Moving forward, future research could focus on addressing the identified challenges and improving the performance of logo detection using YOLO.

REFERENCES

[1] Yuan Liao, Xiaoqing Lu, Chengcui Zhang, Yongtao Wang, and Zhi Tang. 2017. Mutual enhancement for detection ofmultiple logos in sports videos. In Proceedings of the IEEE International Conference on Computer Vision.

[2] Rahul Kumar Jain, Taro Watasue, Tomohiro Nakagawa, SATO Takahiro, Yutaro Iwamoto, RUAN Xiang, and Yen WeiChen. 2021. LogoNet: Layer-aggregated attention centernet for logo detection. In Proceedings of the IEEE InternationalConference on Consumer Electronics.

[3] Tsung Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. 2017. Featurepyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[4] Jonathan Hui, "Understanding Feature Pyramid Networks for Object Detection (FPN)," Medium, 2019. [Online]. Available: https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c.

[5] Hang Su and Guoping Qiu. 2022. Few-shot transfer active learning for logo detection in sports video. SSRN ElectronicJ.

[6] Hichem Sahbi,"Context-Dependent Logo Matching and Recognition"IEEE , MARCH 2013

[7] Hichem Sahbi,"Context-Dependent Logo Matching and recognition"IEEE , MARCH 2013

[8] Chinmoy Biswas "Logo Recognition Technique using Sift Descriptor, Surf Descriptor and Hog Descriptor" (0975 – 8887) Volume 117 – No. 22, May 2015

[9] Remya Ramachandran "Logo Matching And Recognition System Using Surf" Vol 3, Issue 9, September – 2014

[10] Hou, S., Li, J., Min, W., Hou, Q., Zhao, Y., Zheng, Y., & Jiang, S. (2023). Deep learning for logo detection: A survey. ACM Transactions on Multimedia Computing, Communications and Applications.

[11] Bianco, S., Buzzelli, M., Mazzini, D., & Schettini, R. (2017). Deep learning for logo recognition. Neurocomputing, 245.

[12] Rupali R.Sontakke, Lalit B. Damahe "Logo Matching and Recognition: A Concise Review" IEEE 2016

[13] Olorunshola, O. E., Irhebhude, M. E., & Evwiekpaefe, A. E. (2023)A Comparative Study of YOLOv5 and YOLOv7 Object Detection Algorithms