README.md

Logo Detection Project Report :

Authors: Luis Chion (lmchion@berkeley.edu), Eric Liu (eliu390@berkeley.edu), Viswanathan Thiagarajan (viswanathan@berkeley.edu), John Calzaretta (john.calzaretta@berkeley.edu)

Instructor: Allen Y. Yang

Date: 12/09/2022 (MIDS 281 - Fall 2022)

Table of contents

Overview

These days brand logos can be found almost everywhere from images produced by IoT devices (cars and survellaince cameras) to social media postings (Facebook, Tiktok, Instagram). As such, logo recognition is a fundamental problem for computer vision and can be used in the following applications:

- Copyright and Trademark compliance : to detect patent infrigment by identifying logo patterns and colors from a well recognized brand
- Brand related statistics : to understand consumer for targeted advertising. Brand protection, recomendation and identification
- Intelligent traffic-control systems : to recognize a specific symbol like stop or yield sign using camera feed from vehicles

- Document categorization : to detect logo appearing in documents and use it as a statement of ownership

Logo recognition can be considered a subset of object recognition. In general, the majority of the logos are two dimensional objects containing stylized shapes, no texture and primary colors. In some cases, logos can contain text (i.e. Fedex logo) or can be placed in different surfaces (i.e. Coca-Cola bottle or Adidas shoes). The logo detection process can be split in two tasks: determining the location of the logo (bounding box) and logo classification. Finding the logo in a real world image is a challenging task. It is desirable to have a incremental logo model learning without exhaustive manual labelling of increasing data expansion [5]. Also, logos can appear in highly diverse contexts, scales, changes in illumination, size, resolution, and perspectives [6]

To prepare data for classification, we applied several pre-processing methods including Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm, data augmentation and class balancing.

To classify logos, we built three 1-vs-all linear SVMs models and used a YOLO model :

1. Bag of Words SIFT. This model was the standard prior 2015 and has good performance.
2. General Feature Extraction Model. We referred to Choras et al.[2] work to extract color, texture and shape features from images.
3. Mixed model with a combination of Bag of Words SIFT and General Feature Extraction Model
4. YOLO version 7. Top performer among DL models for small and large datasets along with MFDNet and OSF-Logo. Also, quite fast during training and detection.

For the manual models (1-3), we only concentrated on the second task of the logo detection process which is the classification algorithm. Given that the complexity of techniques to identify a logo in an image, we did not think it was feasible to complete the implementation within the course timeline. We used the ground truth bounding boxes provided with the Logos-32plus dataset as our starting point. The bounding box coordinates were used for training both manual and YOLO models. The full image was used in YOLO model during inference without providing the bounding box coordinates. During inference in the manual model, the extracted bounding box was used for predictions.

Related work

In recent years, many datasets have been created to be used as quantitative and qualitative comparisons and benchmarking for logo detection[5]. These sets vary in sizes and contain a realistic set of images with accurate ground truth annotations. Datasets could vary quite a bit in the number of classes and images per class. From FlickrLogos-32 with 32 classes and 2.2K images to PL8K with 8K classes and 3M images. FlickrLogos-32 comprises of images from the real world, and many contain occlusions, apearance changes, and lighting changes. Large datasets such as PL8K have greater class imbalance since images are collected semi-programatically. The images in PL8K span across different enterprises, namely Clothing, Food, Transportation, Electronics, Necessities, Leisure, Medicine, Sports and Others.

#Scale	#Datasets	#Logos	#Brands	#Images	#Objects	#Public	#Year
Small	BelgaLogos [12]	37	37	10,000	2,695	Yes	2009
	FlickrLogos-27 [19]	27	27	1,080	4,671	Yes	2011
	FlickrLogos-32 [13]	32	32	2,240	5,644	Yes	2011
	MICC-Logos [10]	13		720	-	No	2013
	Logo-18 [14]	18	10	8,460	16,043	No	2015
	Logos-32plus [20]	32	32	7,830	12,302	No	2017
	Top-Logo-10 [21]	10	10	700	-	No	2017
	Video SportsLogo [22]	20	20	2,000	-	No	2017
	VLD 1.0 [23]	66	66	25,189	-	No	2019
	SportLogo [24]	31	31	2,836	-	Yes	2020
	VLD-45 [25]	45	45	45,000	-	No	2020
Medium	Logo-160 [14]	160	100	73,414	130,608	No	2015
	Logos-in-the-Wild [26]	871	871	11,054	32,850	Yes	2017
	QMUL-OpenLogo [15]	352	352	27,083	-	Yes	2018
	PL2K [27]	2,000	2,000	295,814	-	No	2019
	FoodLogoDet-1500 [16]	1,500	-	99,768	145,400	Yes	2021
Large	Open Brands [28]	1,216	559	1,437,812	3,113,828	No	2020
	LogoDet-3K [17]	3,000	2,864	158,652	194,261	Yes	2020
	PL8K [18]	7,888	7,888	3,017,146	-	No	2022

Table.1 - Statistics of existing logo detection datasets

Prior 2015, logo recognition has used keypoint-based detectors and descriptions such as Bag of quantized SIFT features (BofW SIFT) and Histogram of Gradients (HOG) [1]. BofW SIFT is a method in which a histogram of visual words are created by bucketing SIFT keypoints using Kmeans clustering algorithm to create the visual words. After this, 1-vs-all linear classifier like SVM is used for logo recognition. In addition to BoW SIFT, Romberg and Lienhart [3] use a feature bundling technique where individual local features are aggregated with features with their spatial neighborhood into bundles. Also, Romberg et al [3] propose to index the relative spatial layout of local features on logo regions by means of a cascaded index.

Choras et al. [2] use general features extraction methods that are application independent such as color, texture, and shape. These features can be further divided into pixel-level features such as color and location, local features as a result of subdivision of the image band in segments or edges, global features over the entire image or sub-area. To represent color features, the author uses color moments (mean, variance, skewness) extracted from RGB, HSV and YCrCb color histograms. Texture is also a powerful descriptor but on its own may not have the capability of finding objects. Texture can be calculated using different methods such as Fourier power spectra, co-ocurrence matrices, SPCA, Tamura features, and Wold decomposition. Next, Shape is one of the primitive features for image content description but measuring is difficult. Shape methods can be divided in region and contour based. While Region-based use the whole area, Contour-based use only info from the contours of an object. For classification, lower order moments are calculated to extract Shape features. The most common moments are geometrical, central, moment invariants, Zernike moments, Hu moments and Legendre moments.

In recent reasearch, deep learning has emerged as the default standard for logo detection. Deep learning models are classified into 4 categories: Convolutional Neural Network models, YOLO-based models, Single Shot Detector-based models and Feature Pyramid Network-based models.

1. R-CNN is a typical proposals-based approach but it is slow in detecting objects due to its Selective Search (SS) algorithm. To overcome this, a Faster R-CNN algo was proposed that uses region proposal network (RPN) to generate region proposals.
2. In contrast to R-CNN, YOLO proposes the use of an end-to-end neural network that makes prediction of bounding boxes and class probabilities all in one

single stage. At a high level, YOlO divides the images into N grids of dimension SxS. These grids predict the bounding boxes along with the labels and the probability of the object being present in a cell. YOLO uses Non Maximal Supression method to exclude the bounding boxes with lower probability scores.

3. Single Shot Detector-based models (SSD) uses multi-scale feature maps to detect objects at different scales. It is comparable to YOLO in that it takes only one shot to detect multiple objects present in an image using multibox. Outperforms a comparable state-of-the-art Faster R-CNN and is widely used in vehicle logo detection.
4. Feature Pyramid Network (FPN) uses a multiple feature map layers similar to SSD. It is composed of two pathways: bottom-up and top-down. The bottom-up is the usual convolutional network for feature extraction. As we go up on the layer, the spatial resolution decreases but the semantic value increases. In comparison, SSD only uses top layers of bottom-up pathway for prediction whereas FPN provides a top-down pathway to construct higher resolution layers from a semantic rich layer.

The tables below show State-Of-The-Art Mean Average Precisions (mAP) for small, medium, and large datasets. We can observe that mAP decreases with the size of dataset. Interestingly, for medium and small datasets, Scaled YOLOv4 performs better than the more conventional Faster R-CNN.

Small dataset performance (FlickrLogos-32)

Method	mAP(%)	Year
FRCN [46]	74.4	2015
BD-FRCN-M [60]	73.5	2016
Video Logo Detector [22]	78.6	2017
Faster RCNN+VGG16_D2_M3 [61]	90.3	2017
Faster RCNN+CAL [15]	74.9	2018
Deep Saliency Map [81]	75.8	2020
Logo-Yolo [17]	76.1	2020
LogoNet [83]	82.2	2021
OSF-Logo [76]	87.0	2021
RetinaNet [59]	40.0	2021
RCNN [59]	65.0	2021
Faster RCNN [59]	74.0	2021
Scaled YOLOv4 [80]	80.4	2021
MFDNet [16]	86.2	2021

Medium dataset performance (QMUL-OpenLogo)

Method	mAP(%)	Year
YOLOv2+CAL [15]	49.2	2018
Faster RCNN+CAL [15]	51.0	2018
Logo-Yolo [17]	53.2	2020
Faster RCNN+CAL+MPCC [85]	49.4	2021
OSF-Logo [76]	53.3	2021
Scaled YOLOv4 [80]	61.9	2021
MFDNet [16]	51.3	2021

Large dataset performance (Open Brands)

Method	mAP(%)	Year
Brand Net (SMA) [28]	60.4	2020
Faster RCNN+RFS+MST [5]	64.6	2021
Cascade RCNN+Soft NMS [6]	65.1	2021
Green Hand [8]	65.5	2021
Cascade RCNN+DCN v2 [7]	70.2	2021

Robust and accurate detection is still difficult. Logos tend to be small in size and maybe difficult to detect them in complex backgrounds (i.e. logo sign on a busy street). The backgrounds can be very diverse in nature. Logos can be on bottles, shirts, cars, billboards and can be of different textures (i.e. nike logo in shoes and clothes). Sub-branding detection can impose additional difficulties when there are subtle differences between parent brands and sub-brands (i.e. coca-cola and diet coke). There is still plenty of work to do to improve logo detection. Higher resolution feature maps have been used with sucess but it is computationally very expensive and too slow for real-time applications.

Some future research directions include: Lightweight logo detection to reduce model complexity while mantaining the same accuracy. Weekly supervised logo detection to automate the annotation not only reduces the cost but also improves the generalization and robustness of the model. Video logo detection which provides more information to businesses by adding correlation between consecutive images. Tiny logo detection with not enough pixel information for recognition. Long tail logo detection for growing small businesses where you cannot find enough image samples. Incremental logo detection where we assume an open dataset where new logo and logo variations emerge every day.

Dataset

Logos-32plus is a collection of 12,312 real-world photos that contain 32 different logo classes. It is an expansion on the the FlickrLogos-32 dataset. Both has the same classes of objects but Logos-32plus has substantially more images. Logos-32plus is designed to be more representative of the various conditions that logos appear in and more suitable for training keypoint-based approaches to logo recognition. Logos appear in high contrast on approximately planar or cylindrical surfaces with varying degrees of obstruction. To construct this dataset, images were scraped from Flickr and Google Images through a text-based search on image tags by the owners. To increase variability in the data distribution, different queries were put together by concatenating a noun plus the logo name (i.e. "merchandising Becks", "can Becks", "drink Becks"). Scraped images were manually filtered by the owners to remove unfocused, blurred, noisy, or duplicate images [1].

We selected 10 logo classes and corresponding images from the Logos-32plus dataset to use in our project for training and evaluation. Each image in this dataset is labeled with a single class, and the 10 classes each contain 300 photos on average. Bounding box annotations are provided for each occurrence of a logo in an image; photos may have one or multiple instances of the logo corresponding to the labeled class. In cases where an image contains logos belonging to multiple classes, only logos corresponding to the image class are annotated. The distribution of images and bounding boxes per class are shown in Fig. 1. Note that bounding box counts are significantly larger than image counts due to images containing multiple occurrences of a logo.

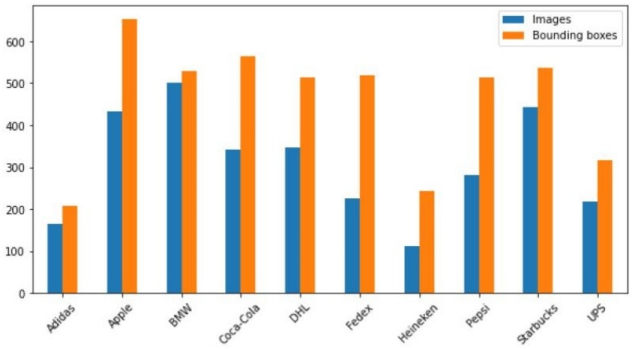


Fig.1 - Distribution of logo images and bounding boxes per class

Methods

Non-deep learning methods to logo prediction typically involve two approaches. The first one is logo localization, which identifies potential locations in an image where a logo may be present. This can be solved by using a correlation tracker to detect image regions that have high correlation with a mask image. Multiple mask images are produced from each logo via affine transformation, rotation, and resizing. Regions that produce a sufficiently high correlation with any mask image are annotated with a bounding box and the class of the mask image. The second one is logo classification, which uses a feature-based model to perform image recognition on the bounding boxes produced by a logo localization model or a matching algorithm. Since the Logos-32plus dataset provides ground truth bounding boxes for all images, this project assumes a strongly-labeled dataset as input and focuses only on the logo classification problem within the bounding box.

Based on the literature review and the performance of YOLOv4 compared to R-CNN for this type of problem we selected YOLO. The YOLO deep learning model takes entire images with annotations as input, while our manual classification model takes labeled bounding boxes as input. The train-validation-test split is applied at the image level before preprocessing. We apply a 70-15-15 train-validation-test split in order to maximize the size of the validation and test sets to increase the generalization and robustness of the model.

Image Preprocessing

Using the ground truth bounding boxes provided by the Logos-32plus dataset, all bounding boxes are extracted from each image. Each bounding box is now considered a unique logo image that belongs to the same class and data split as the source image. Image contrast normalization is performed on by applying the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm with 4x4 tile size to the luminance channel of each image. Example output of this step is shown in Fig. 2. Data augmentation generates additional training examples from a single image by applying random 3D rotation transformations and color inversions. Class balancing is enforced by adjusting the number of generated images such that the final image counts are uniform.

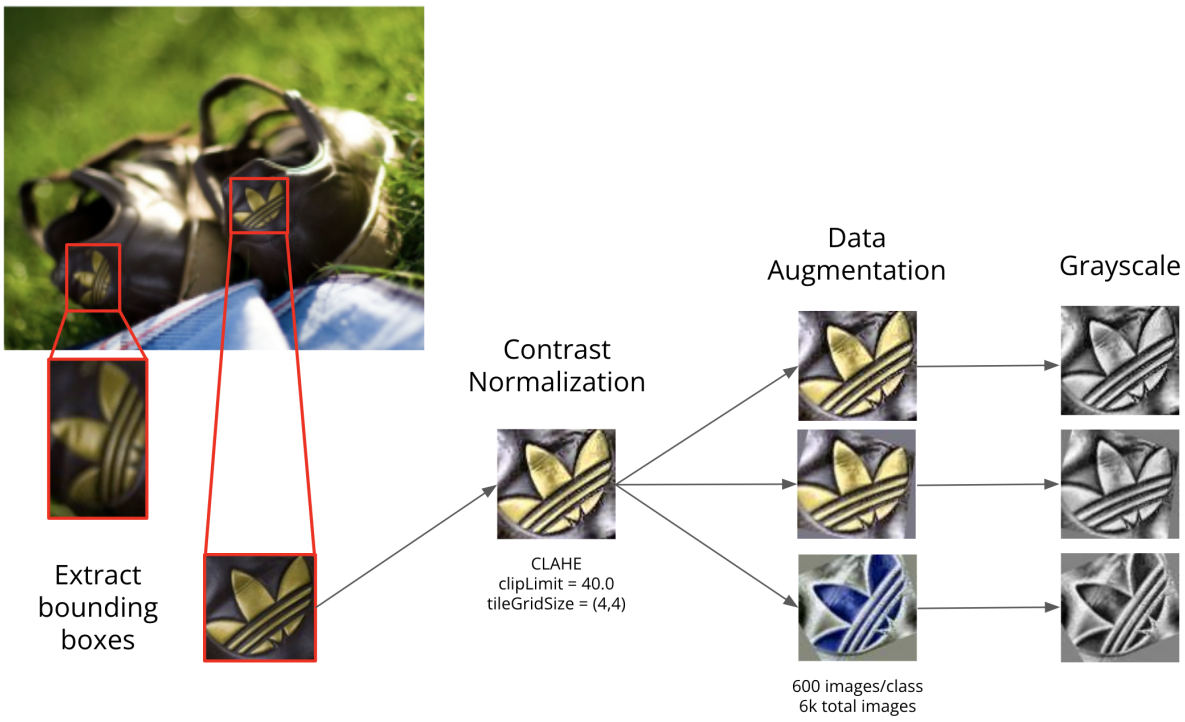


Fig.2 - Data Augmentation Example using Adidas image

Bag of Words SIFT (BoW SIFT)

Our dataset consists of real world photos with different logos in them. The logos found in the images are having different colors, scales, illumination, rotations, local affine distortions and partial occlusion. One of the features that could work well with our classification task is SIFT (Scale-invariant feature transform). We selected SIFT for the manual model as it is rotation and scale invariant and has the potential to work well when compared to other existing descriptors when there are distortions as described above in the dataset.

SIFT or Scale Invariant Feature Transform is a feature detection algorithm in Computer Vision. SIFT locates features in an image, known as "keypoints". Keypoints are scale, noise, illumination and rotation invariant. Another important characteristic is that the relative position between the features does not change from one image to another. Each keypoint is a 128-dimensional feature descriptor (when 1 layer is used). A vocabulary is formed by sampling features (or keypoints) from the training set and clustering them using K-means algorithm. This process partitions 128 dimensional SIFT features space into N number of regions and allow us to create histograms of visual words. After that, the SIFT histogram gets normalized and a classification model is trained using SVM. A detailed description of how SIFT was implemented is described below.

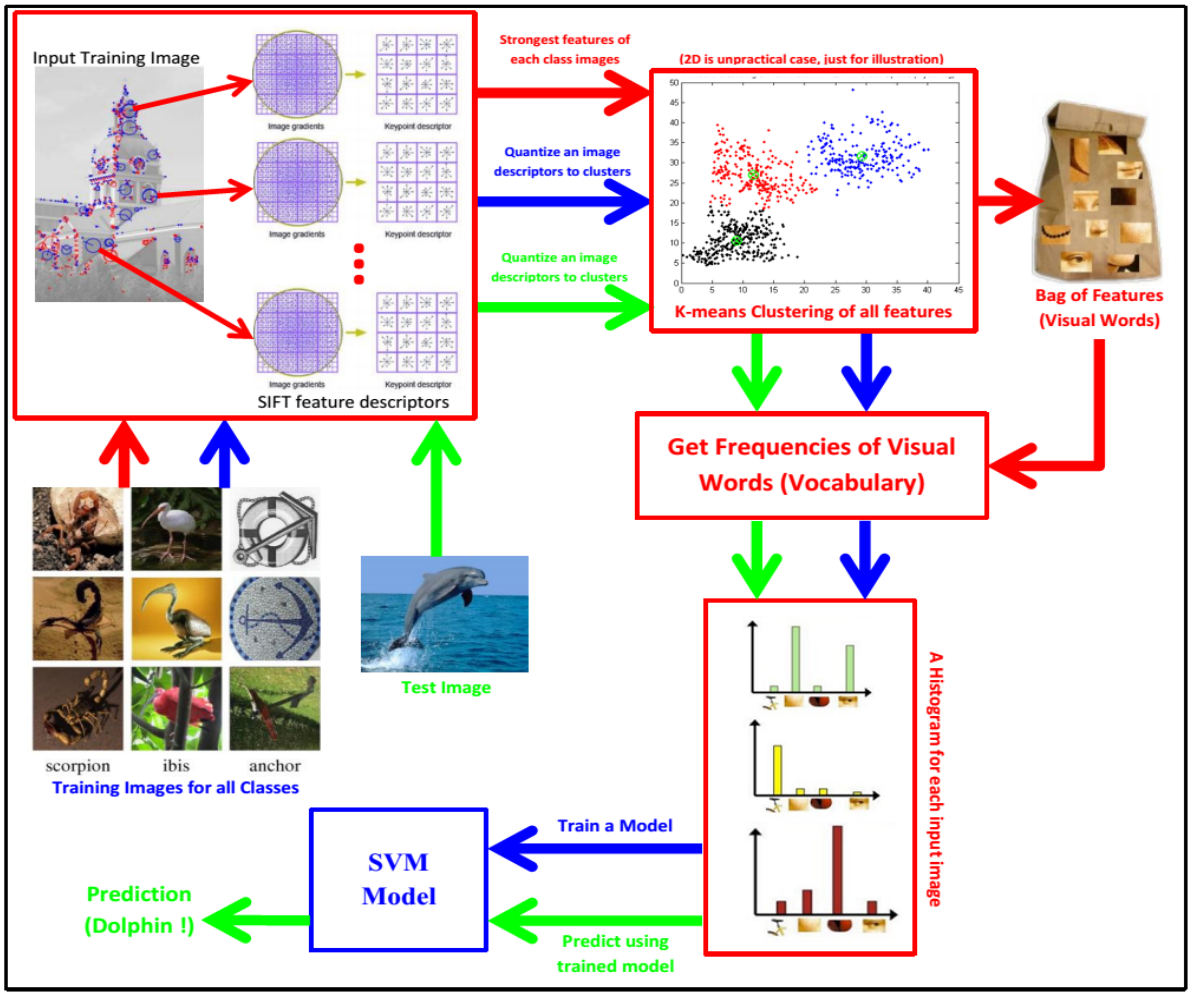


Fig.3 - Bag of Words SIFT diagram from https://heraqi.blogspot.com/2017/03/BoW.html

Our step by step implementation of SIFT and classification algorithms on the logos dataset can be accessed from this notebook. The notebook has two sections. In the first section, the logos were extracted from the images using manual coordinates and processed through SIFT feature extraction, histogram build, and training with no data pre-processing. In the second section, the input bounding boxes were already pre-processed and augmented on which SIFT features extraction, histogram build, and training was performed. In both sections the images used converted to grayscale.

General Feature Extraction (GFE)

Along with the BoW SIFT features, additional non-learned features were extracted from the image bounding boxes, these include: Shape, Color and Texture feautures. The motivation is to understand if model performance improves when fit on BoW SIFT and additional features.

Shape

Image moments are used to describe the shape of an object in an image. These moments are described in the 1962's Ming-Kuei paper [3] and capture information like the area of the object, the centroid and the orientation. Hu moments should not be used in situations where there is noise, occlusion or a lack of clean segmentation since it is very hard to get a dependable and repeatable centroid. Hu moments are invariant to translation, scale, rotation and reflection. To calculate Hu moments, we first used Canny to calculate the edges of a log with threshold 1 and 2 set to 100 and 200 respectively. After, Hu moments are calculated as follows:

The regular moment of a shape in a binary image is defined by:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

where I (x, y) is the pixel intensity value at the (x ,y)-coordinate.

In order to obtain translation invariance, we need to take our shape measurements relative to the centroid of the shape. The centroid is simply the center(x ,y)-coordinates of the shape, which we define as x– and y– respectively.

With the centroids, we can compute relative moments which are centered about the centroid:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y)$$

These relative moments do not have much discriminative power to represent shapes, nor do they posses any invariant properties. To solve that, Hu took these relative moments and constructed 7 separate moments which are suitable for shape discrimination:

$$M_1 = (\mu_{20} + \mu_{02})$$

$$M_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2)$$

$$M_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{30})^2$$

$$M_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$$

$$M_5 = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})((\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2) + (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})(3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2)$$

$$M_6 = (\mu_{20} - \mu_{02})((\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2) + 4\mu_{11}(\mu_{30} + 3\mu_{12})(\mu_{21} + \mu_{03})$$

$$M_7 = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})((\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2) - (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})(3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2)$$

Color

Color features are advantageous in image classification as they are robust to rotation, simple to compute, and require minimal storage. Color features can be effective in the logo domain as many brand logos use consistent color schemes the majority of the time. However, a key challenge is presented when a brand uses several different color schemes for a single logo. For example, the classic Coca Cola can is known for its red and white color scheme, but the Coca Cola Zero product uses black and red coloring.

When extracting color features, the standard approach is to build a color histogram to represent an image. Given a color histogram, color moments are an effective way to condense the representation. For each image and color model (RGB, HSV, YCrCB), we extract the first (mean), second (variance) and third order (skewness) color moments, thus resulting in nine color moment features per image.

For each color component *k*, the first color moment is defined by

$$M_k^1 = \frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} f_k(x, y)$$

where the color of the pixel (*x, y*) is represented by $f_k(x, y)$. For the second and third moments (*h = 2, 3*), each color component *k* is defined as

$$M_k^h = \left(\frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} \left(f_k(x,y) - M_k^1 \right)^h \right)^{\frac{1}{h}}$$

Texture

Texture is another important property of a logo; while it does not uniquely describe a logo, it may help differentiate textured logos from non-texture ones. For example, FedEx logos are typically found in print (plastic) or on vehicle decals (metal), and Adidas logos are frequently found on clothing and shoes (cotton and leather).

A statistical approach to generating texture features uses a gray level co-occurrence matrix, which is a histogram of co-occurring grayscale values (i, j) at a given distance d (defined in polar coordinates) over an image. The co-occurrence matrix C(i, j) is defined by

$$C(i,j) =$$
$$= card \left\{ \begin{array}{l} ((x_1,y_1),(x_2,y_2)) \in (XY) \times (XY) \\ for \quad f(x_1,y_1) = i, f(x_2,y_2 = j \\ \\ (x_2,y_2) = (x_1,y_1) + (d\cos\theta, d\sin\theta); \\ for \quad 0 < i,j < N \end{array} \right\}$$

where card indicates the number of elements in the set.

A set of features can be extracted from the co-occurrence matrix to reduce the dimensionality of the feature space. Figure 3 captures the set of features extracted from C(i,j)

$$Energy = \sum_i \sum_j C(i,j)^2$$

$$Inertia = \sum_i \sum_j (i-j)^2 C(i,j)$$

$$Correlation = \frac{\sum_i \sum_j (ij) C(i,j) - \mu_i \mu_j}{\sigma_i \sigma_j}$$

$$DifferenceMoment = \sum_i \sum_j \frac{1}{1+(i-j)^2} C(i,j)$$

$$Entropy = -\sum_i \sum_j C(i,j) \log C(i,j)$$

Fig.3 - Formulas for Texture Features from Co-Occurrence Matrix (from **CITE PAPER**)

Our implementation of these non-learned features can be found in general_feature_extraction.ipynb.

Mixed Models from Non-Learned Features

With the non-learned features described above, we fit a set of models on each of three feature sets: BoW SIFT only, GFE only (shape, color texture), and the combination of BoW SIFT and GFE. The goal of this exercise is to understand how effective are each of the feature sets, and if any performance lift is achieved by mixing them.

For each feature set explored, we selected Support Vector Machine algorithm for classification based on Romberg et al. work. The validation set was used to tune the hyperparameters of each model. The model configurations are displayed in Table 2 below.

Model Name	Model Form	Feature Set	Tuned Parameters
Mixed SIFT + GFE	SVM	BoW SIFT + Shape, Color, Texture features	- Penalty: L2 - C: 0.01 - Class Weights: None
BoW SIFT	SVM	BoW SIFT only	- Penalty: L2 - C: 0.25 - Class Weights: Balanced
GFE Model	SVM	Shape, Color and Texture features only	- Penalty: L2 - C: 0.0001 - Class Weights: None

Table 2 - Top Model Configuration per Feature Set

The implementation and evaluation of each model in Table 2 can be found in gfe_modeling.ipynb. The results of each model are displayed and discussed in the results section.

YOLO

YOLO is a deep learning algorithm for real-time object detection. It is capable of not only providing the location of an object in an image (bounding boxes) but also identify the object (classification). It uses a combination of approaches such as fast R-CNN, Retina-Net and Single-Shot MultiBox Detector (SSD). YOLO has become the standard in object recognition due to its speed, detection accuracy, good generalization and the fact that it is open-source.

In preparation for the training, we loaded 3,062 groundtruth images into Roboflow using the same train test validation partition (70%-15%-15%). After uploading, images were auto-oriented using the correct EXIF orientation stored in the metadata. Also, images are resized to 640x640 pixels to improve training performance. No further data augmentations were applied.

For training YOLO, we used a batch size of 16 and with 300 epochs as recommended in Tips for Best Training Results Documentation. For testing, we set our confidence and IOU threshold equal to 50%.

The step by step implementation of YOLO can be found in Yolov5.ipynb. These notebooks are created from Roboflow's Blog: How to Train YOLOv7 on a Custom Dataset. It shows step by step how to download the dataset, custom train and run evaluations.

Results and Discussion

In this section, we summarize and compare the performance of different models. We also discuss the performance of each model in greater detail using class-level metrics. In the end, we examine specific images that were not predicted correctly by the top performing model to better understand the model's issue and areas of potential improvement.

Overall Performance

Table 3 compares the performance of the YOLO model to the different GFE models trained on non-learned features.

Model Name	Model Form	Feature Set	Accuracy	Precision	Recall	F1
YOLO V7	YOLO V7	Learned by Model	0.88*	0.88	0.89	0.88
Mixed SIFT + GFE	SVM	BoW SIFT + Shape, Color, Texture	0.89	0.88	0.87	0.88
BoW SIFT	SVM	BoW SIFT	0.80	0.80	0.81	0.80
GFE Model	SVM	Shape, Color, Texture	0.79	0.75	0.74	0.74

Table 3 - Comparison of model performance *YOLO models use mAP@0.5 for accuracy metric

It was expected that the YOLO model would be a strong performer as it is known to deliver state of the art results across many image classification tasks. Whereas the GFE models are fit on deterministic features extracted from images, the YOLO model is able to learn abstract features in fine tuning that apply specifically to the logo domain. However, the Mixed GFE model is competitive with YOLO, which highlights the signal provided by the BoW SIFT and other non-learned features.

The top performing model that uses only non-learned features is the Mixed GFE model which was comparable to YOLO performance. We hypothesized that the BoW SIFT features alone would deliver the top performance; however, the additional non-learned features provided additional signal. The union of all non-learned features makes the model more complex and the fact that improved performance tells us that the SIFT model is not complex enough and underfit the data. It is important to keep in mind that BoW SIFT Bag of visual words were limited by compute constraints and could have increased complexity itself with more than 1500 words, aside from the other GFE features.

Class-Level Performance

In multi-class classification problems, it is crucial to understand the model performance across classes, rather than just the aggregate measures. Table 3 shows the class-level performances of the YOLO, Mixed GFE, and BoW SIFT models.

Class	YOLO V7	Mixed GFE + SIFT	BoW SIFT
Adidas	0.79	0.67	0.63
Apple	0.92	0.79	0.66
BMW	0.95	0.90	0.85
Coca cola	0.73	0.85	0.76
DHL	0.96	0.96	0.83
FedEx	0.99	0.90	0.89
Heineken	0.80	0.89	0.85
Pepsi	0.79	0.90	0.70

Class	YOLO V7	Mixed GFE + SIFT	BoW SIFT
Starbucks	0.98	0.98	0.98
UPS	0.92	0.93	0.86

Table 3 - Comparison of F1 score per class across models

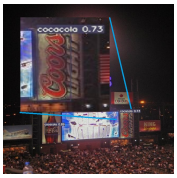
Looking across logo classes and models, there are some classes that each model does well on, and some with which each model struggles. Each model has a strong F1 score for Starbucks, UPS and Fedex classes; UPS and Fedex are both text based logos with distinct edges, while Starbucks is highly distinct and likely easier for models to differentiate. Adidas and Coca Cola have F1-scores mostly below or near 0.8 for each model; this could be due to the fact that these two brands have greater diversity in the appearance of logos across different images and have larger homogeneous area.

Comparing YOLO to the Mixed GFE & SIFT models, YOLO performs markedly better for Apple and BMW, and markedly worse for Coca Cola, Heineken and Pepsi. One explanation could be that the former classes are often found on flat surfaces, and the latter are all drinks which are typically found on curved surfaces. Additionally, research has shown that YOLO models perform well on smaller objects and in low contrast environments, which apply closely to Apple and BMW [8].

Comparing Mixed GFE & SIFT model to standalone SIFT model, the Mixed model provides a strong lift in each of the classes where BoW SIFT struggles: Apple, BMW, and Pepsi. One explanation could be that each of these logos have few corners and thus SIFT might struggle to consistently identify keypoints. The color, shape and texture features add more signal when detecting keypoints in the logos that are less consistent. The remainder of classes are roughly comparable, with the Mixed model slightly outperforming the SIFT model.

Image-Level Error Analysis

For the top performing YOLO V7 model, we picked out several examples of misclassified test images to further understand the limitations and weaknesses of the model. This exercise showed that the YOLO model made mistakes more frequently in cases with small, blurry, and occluded images as well as cases of unusual logos that deviate from the common case (such as the Heineken pint image).

Undetected PEPSI with blur	Detected an occluded Starbucks	Shouldn't have detected as Coca-Cola	Not detected small BMW
			
Undetected small Heineken sign	Shouldn't have detected as Heineken	Not detected different Heineken logos	Not detected warped Coca-Cola
			

State of the Art Comparison

As discussed in the Dataset section, our models are developed on a subset of classes from the Logos32+ dataset (due to compute constraints) and also do not include a background class. Though not an apples-to-apples comparison due to the dataset differences, the below Figures 7 and 8 show our models in comparision with the models in the literature that inspired them.

Method	Train Data	Precision	Recall	F1	Accuracy
BoW SIFT (Romberg et al.)	FL32	0.99	0.78	0.88	0.94
BoW SIFT (W281)	L32+ Subset	0.8	0.81	0.8	0.8
Mixed GFE + BoW SIFT (W281)	L32+ Subset	0.89	0.88	0.87	0.88

Fig. 7 - Comparison with BoW SIFT in Romberg et al. [3]

Method	Train Data	Precision	Recall	F1	Accuracy
CNN (Bianco et al.)	FL32	0.91	0.85	0.88	0.88
YOLO V7 (W281)	L32+ Subset	0.88	0.88	0.89	0.88

Fig. 8 - Comparison with CNN in Bianco et al. [1]

The BoW SIFT model fit on FlickrLogos-32 dataset in Romberg et al. shows higher performance than our BoW SIFT fit on the subset of classes in Logos32+ and the Mixed GFE & SIFT model. The CNN model fit on FlickrLogos-32 dataset in Bianco et al. only slightly outperforms our YOLO model fit on the subset of classes in Logos32+ in terms of precision. A direct comparison is challenging without fitting our models on similar data, preprocessing and augmentation. We hypothesize that the differences can be primarily attributed to our models working with a subset of classes, not including a background class, and not utilizing similar data augmentation.

Challenges and Next Steps

During the data pre-processing, augmentation and the model building phases we encountered a few challenges which are listed below:

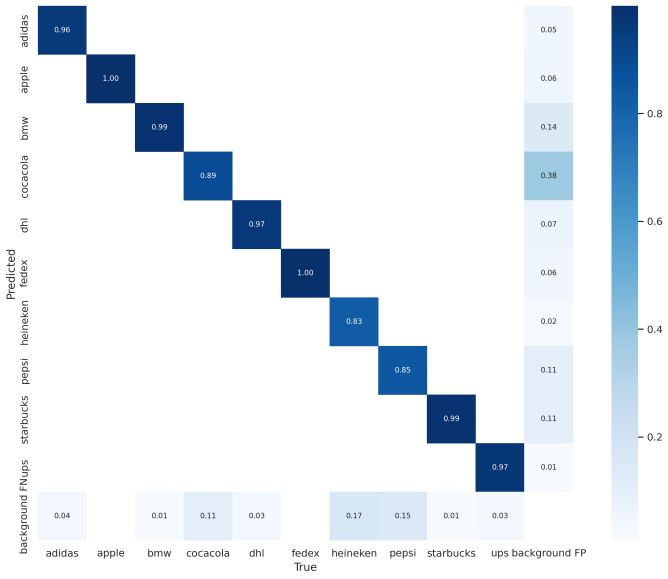
- The Logos 32 plus dataset consists of 32 classes. We had to reduce the selection to 10 classes to keep data processing and training manageable.
- A SIFT Bag-of-visual-words histogram model was used during training. As we increased the number of bins/clusters for the descriptors, the model continued to show improved performance on the validation set. We hit computing capacity before we could further increase complexity and validate the model.
- Contrast Limited AHE (CLAHE) was used to reduce this problem of noise amplification. Due to the use of a smaller kernel in this step, keypoints increased by 3.5 times per image. This caused severe performance bottle necks while running the k-means algorithm to create SIFT Bag-of-visual-words.

- Extracting SIFT keypoints from some of the homogeneous logo images were challenging and no keypoints were detected in some cases. These have been excluded during both training and validation.
- Handling same brand logos with different color schemes, mirror images and textures on different surfaces.

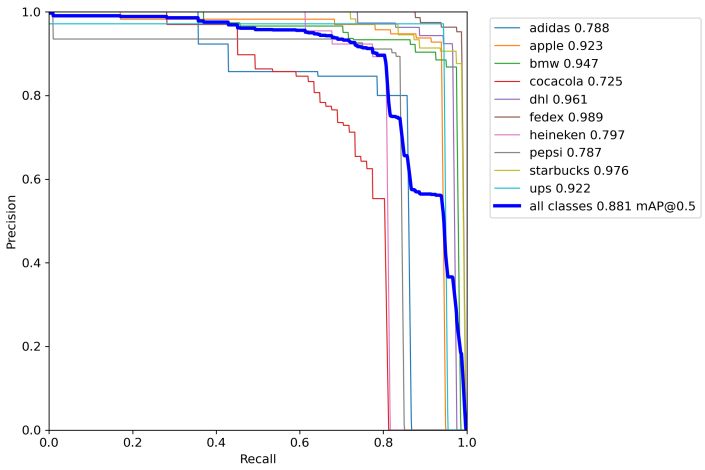
As an extension to this project, we have identified a few next steps:

- Build a classification model that includes all 32 classes
- Perform Contrast Limited AHE with a larger kernel limiting the contrast amplification and the noisy details
- Perform data augmentation for the training similar to the Bianco et al. paper[1]
- Use 2 layers in SIFT with 160 dimension keypoint descriptors
- Increase complexity of the Mixed model by introducing additional features into the model
- Execute YOLO algorithm on the dataset by enabling data augmentation options

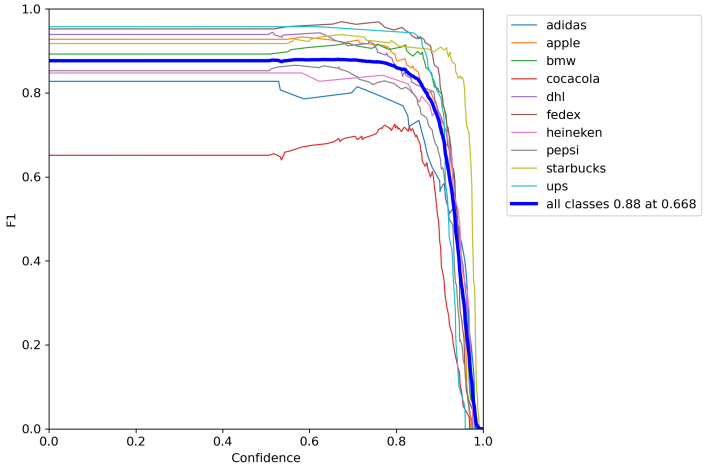
Appendix 1 : Additional YOLO metrics



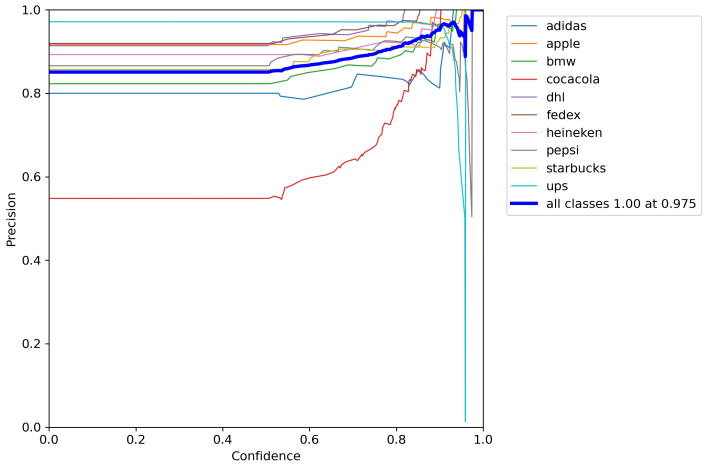
YOLOv7 Test Confusion Matrix



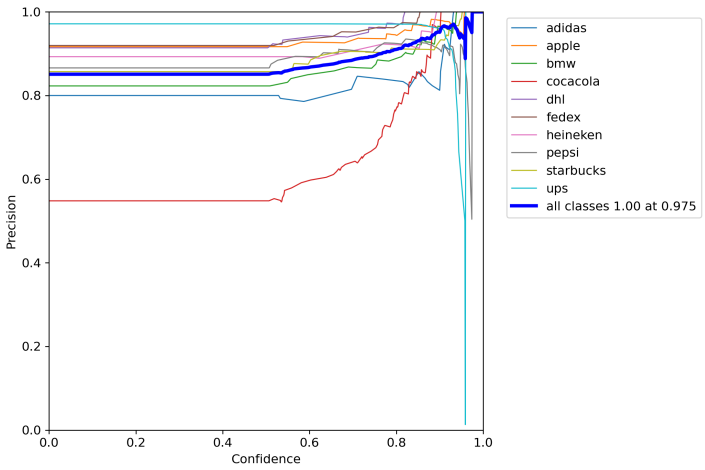
YOLOv7 Test PR Curve



YOLOv7 Test F1 Curve



YOLOv7 Test P Curve



YOLOv7 Test R Curve

If any of the notebooks, links or images are not accessible in the report, please go to the GitHub repo shown and refer to the materials.
https://github.com/jcalz23/logo_detection_w281/

References

[1] Simone Bianco, Marco Buzzelli, Davide Mazzini and Raimondo Schettini (2017). Deep Learning for Logo Recognition. Neurocomputing. https://doi.org/10.1016/j.neucom.2017.03.051

[2] Choras, Ryszard S.. (2007). Image feature extraction techniques and their applications for CBIR and biometrics systems. International Journal of Biology and Biomedical Engineering. https://www.researchgate.net/publication/228711889_Image_feature_extraction_techniques_and_their_applications_for_CBIR_and_biometrics_systems

[3] Stefan Romberg and Rainer Lienhart. 2013. Bundle min-hashing for logo recognition. In Proceedings of the 3rd ACM conference on International conference on multimedia retrieval (ICMR '13). Association for Computing Machinery, New York, NY, USA, 113–120. https://doi.org/10.1145/2461466.2461486

[4] Romberg, Stefan & Pueyo, Lluis & Lienhart, Rainer & Zwol, Roelof. (2011). Scalable logo recognition in real-world images. 25. 10.1145/1991996.1992021. https://dl.acm.org/doi/10.1145/1991996.1992021

[5] Hou, S., Li, J., Min, W., Hou, Q., Zhao, Y., Zheng, Y., & Jiang, S. (2022). Deep Learning for Logo Detection: A Survey. ArXiv, abs/2210.04399. https://arxiv.org/abs/2210.04399

[6] C. Li, I. Fehérvári, X. Zhao, I. Macedo and S. Appalaraju, "SeeTek: Very Large-Scale Open-set Logo Recognition with Text-Aware Metric Learning," 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022, pp. 587-596, doi: 10.1109/WACV51458.2022.00066. https://ieeexplore.ieee.org/document/9706752

[7] H. Su, S. Gong and X. Zhu, "WebLogo-2M: Scalable Logo Detection by Deep Learning from the Web," 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 2017, pp. 270-279, doi: 10.1109/ICCVW.2017.41. https://ieeexplore.ieee.org/abstract/document/8265251

[8] Shuo Yang, Junxing Zhang, Chunjuan Bo, Meng Wang, Lijun Chen (2018). "Fast vehicle logo detection in complex scenes," Optics & Laser Technology Volume 110, 2019, pp.196-210, doi: 10.1016/j.optlastec.2018.08.007. https://www.sciencedirect.com/science/article/abs/pii/S0030399218310715

[9] Ming-Kuei Hu, "Visual pattern recognition by moment invariants," in IRE Transactions on Information Theory, vol. 8, no. 2, pp. 179-187, February 1962, doi: 10.1109/TIT.1962.1057692.