Master Degree Project

Content Extraction from Marketing Flyers

Haripriya Surendran Nair

03. June 2022

Supervisor: Anton Kalén
Examiner: Juhee Bae

Master's Degree Project 15 Credits in Informatics
with a Specialization in Data Science
Spring term 2022

ABSTRACT

Shops, supermarkets, and real estate dealers use marketing flyers in abundance to advertise the weekly and seasonal offers these days. The information available in such flyers is a good source marketing study. However, this information is not recorded in any central repository for future usage. This work focuses on the feasibility of using deep learning technology to detect objects from marketing flyers. The detection and recognition of objects from media files such as images is a prominent computer vision domain. Although previous investigations have used two-stage object detection techniques to solve the problem using models like Faster R-CNN, this work experiments with the usage of the state-of-the-art single-stage object detecting method YOLO in the object detection from marketing flyers. The work utilizes different training techniques of the YOLO algorithm and identifies the best one to use for detecting objects from marketing flyers.

Transfer learning and custom object training are the two methods of training YOLO. Transfer learning uses the pre-trained knowledge, while custom training is done with two different methodologies. One is by using a pre-annotated dataset like google open images. Another option is to collect representative data and manually annotate the object's position in it. Custom training with manual annotation achieved a mean average precision MAP of 98.56%. Hence it shows that single-stage object detection can be used to detect and classify objects from flyers provided to have representative datasets for training the model.

Table of Contents

1. Introduction

A vital area in computer vision is object detection which manages the process of finding, detecting, and categorizing objects from images. The task is under extensive investigation in academia and real-world applications[1]. Deep convolutional neural networks and GPU computing power have contributed notable efforts to the advancements in object detection in recent years[1]. The recent improvements lead to the expansion of object detection in different applications such as object detection from noisy images. Object detection from noisy images is challenging due to the design complexity, inconsistent font, non-uniform positioning, and orientation of elements. But it can lead to the development of real-world applications such as product identification in visual aid applications, traffic signal identification, vehicle and pedestrians identification for autonomous vehicles, medical features identification in healthcare, object identification for applications like google lens, and so on [2].

Marketing flyers, also known as sales flyers, are noisy images with contents. The design composition generally does not follow a single standard rendering approach and hence the information extraction from such documents is challenging [3]. Figure 1 shows two sample marketing flyers. Marketing flyers are a good source of information in several fields. It comprises periodic offer updates at supermarkets, offers of electronic items, event promotions, and real estate ad brochures to name a few. Flyers are in both physical and digital forms. Businesses consider flyers as an advertising tool to increase their visibility in the marketing field due to their low production cost and easy distribution[3]. Identifying products from flyers are of great importance as the information present in flyers are hard to gather from any other means. Details about lack of a centralized system for collecting flyer information is described in section 1.1.

Figure 1 - Sample marketing flyers

1.1 Motivation

Sales flyers are still important in marketing. A study conducted by Data & marketing association (DMA) in 2018, it is found that the flyers still have a surprisingly positive effect on marketing. Their results say 57.1% of receivers open and read the contents[4]. London circular distribution company state that flyer campaigns with a personalized and targeted audience will make them the best way to gain more customers[5].

Marketing analysis can be done with the data gathered from flyers. It can act as a searchable collection of deals from online and local markets and can be used for potential applications such as price comparison shopping engines or a shopping assistance application in which users can search for a product, and get a list of nearest shops sorted based on the product price [6] According to a study[7], the commercial real estate industry in the US does not have a centralized database or an established source of information. Several commercial real estate inventories collect commercial real estate data using information from flyers, contacting brokers, or visiting physical sites. Having an established information source would be a benefit and automatic information extraction will aid in data collection for such endeavors. Visual aids are another potential high-demand application that can be derived from the collected information [7], [8].

Product images are a vital part of marketing flyers and to extract information with more accuracy and speed, a combination of text and object detection is required [3]. It is wise to check out the latest advances in object detection in marketing flyers to see how well the new technologies operate.

1.2 Problem definition

Deep learning techniques are already implemented in object detection from noisy images. Most of the previous works[3], [6], [7], [9] based on deep learning techniques in object detection from marketing flyers used two-stage detection where images are processed twice to first propose the candidate object bounding boxes and then features are extracted to classify the object [1]. There is another technique of object detection known as single-stage object detection where an image is passed only once through the network and proposes the predicted boxes from input images directly without the region proposal step, thus they are time efficient and can be used for real-time devices[1]. Previous studies achieved adequate accuracy of 90% with two-stage detectors[3] and so this study aims for better accuracy. On this basis, the following research topic arises:

- Can dataset of individual objects be used for training single-stage object detector like YOLO for detecting objects from marketing flyers containing number of objects?
- How do combine different training methods of YOLO to detect objects from marketing flyers?

For the scope of this study, only the products categories of fruits and vegetables are considered as that are the most occurring items in marketing flyers.

2. Background

Both text and object detection methods are used for content extraction in the previous papers [3], [6], [9]. Older studies[6], [7], [9] used marketing flyers in PDF format and converted them to HTML and used the document object model elements to extract tokens and classified them using random forest and CNN classifiers. Later deep learning technologies are introduced and approached the problem as a computer vision problem for text and object detection in images. Faster R-CNNs are the most used deep learning methods in previous papers [3], [6], [9].

In the paper[3] by Mosquera & Genc have proposed a merge algorithm to merge the text detection and object detection modules. They have used Faster R-CNN for product detection, Bi-directional LSTM and Microsoft API for text detection, and a merge algorithm to merge the results based on the distance between the elements and classify. Another paper[6] by Gallo et al. deals with similar data in PDF format to identify information from physical flyers and advertising emails. They have converted the PDF documents to HTML format and HTML elements are grouped to identify different classes such as product image, price, title, description, and other texts. But they have used Random forest classifier for extracted token classification. Later studies in the field[9] used Faster-RCNN for classification. A more generalized approach is done in paper [10] for information extraction from heterogeneous visually rich documents. They proposed a segmentation algorithm that decomposes a visually rich document into a group of logical blocks that are visually isolated but semantically coherent areas in the document. Then a supervised search-and-select method for identifying the named entities within these documents by utilizing the context boundaries defined by these logical blocks.

Another approach [11] is to make use of pre-trained models and fine-tune them for our needs with the transfer learning technique. Transfer learning is the process of applying features learned on one problem to a new, related problem[12]. This can be applied to the problem as it has multiple categories of images and can benefit from pre-trained models to identify some categories further with more specific training on problem-specific data, better results can be achieved.

Although previous studies have acquired acceptable results, all of them follow the two-stage deep learning object detection approach where the object detection is accomplished in two stages. The first stage is the extraction of the region of objects - finding out the possible location of the object and the second stage is to identify and classify the object. The two-stage detectors are known to be powerful but relatively slow, and the computational cost is higher [13]. Another approach to object detection is single-stage detection where only a single evaluation is happening to localize and classify the object in an image. It is a state-of-the-art, real-time object detection system that offers extreme levels of speed and accuracy [14].

SSD - Single Shot Detector and YOLO - You Look Only Once are the two mainstream single-stage object detectors [1]. According to a study [15], the YOLO model performs better than SSD with a high MAP (Mean Average Precision) of 80.17% and has advantages in detection speed as well as training time. In a comparison study between YOLO and SSD [16], YOLO performs well for smaller object detection. Roboflow, a computer vision application development company identifies YOLO as the best real-time object detector in computer vision in their blog[17]. The key difference between YOLO from other object detectors is in treating the detection task as a single shot regression approach for identifying bounding boxes. They also termed that YOLO models are often very fast and very small making them faster to train and easier to deploy, especially to compute-limited edge devices.

Considering these points, YOLO is taken as the detection model for identifying products from flyer images as the product objects are smaller in size in such flyers, and for real-time applications, speed is more important. YOLO is open source and is built on top of a neural network framework for training and testing computer vision models called darknet[18] in 2016 by Rodemon et al. [19].

YOLO

YOLO requires a single neural network to do both detection and prediction of bounding boxes for identified entities, which was earlier a multi-step approach. As a result, it is steadily calibrated for detection performance and can identify and classify objects significantly faster than two individual neural networks. It accomplishes this by utilizing traditional image classifiers for the regression job of determining object bounding boxes. The concept behind YOLO is to split an image into smaller images. The image is divided into a square grid as shown in figure 2. Each grid cell predicts a single instance. The cell in which the object's center is located is responsible for forecasting that object. The red grid cell, for example, attempts to forecast the "dog" item whose center lies within that grid cell[20].

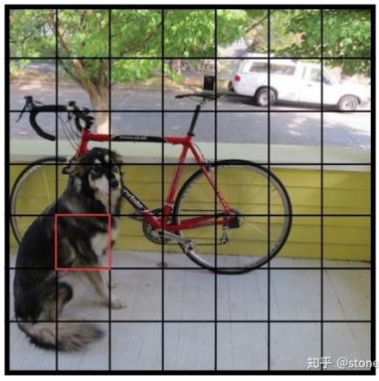

Figure 2 – Image split into grids. Red grid tries to predict the dog[20].

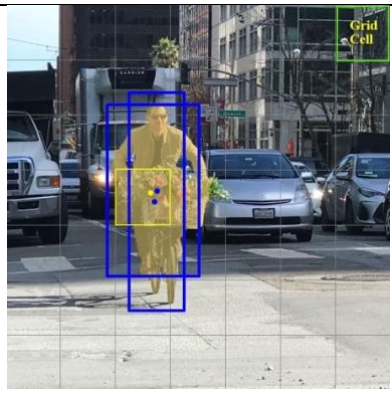

Figure 3 - Grid cell predicts bounding boxes [20]

A fixed number of bounding boxes is predicted by each grid cell. The yellow grid cell in the figure 3 makes two bounding boxes predictions (blue boxes) to determine where the person is[20]. Each cell will forecast B bounding boxes as well as a confidence score for each of them. Each of these bounding boxes is made up of five numbers: the x and y coordinates, width, height, and confidence. The coordinates '(x, y)' denote the location of the estimated bounding box's center, while the width and height are fractions of the total image size. Depending on the presence of an object in the cell, the confidence score varies from 0.0 to 1.0. If there is no item in a cell, the value should be 0.0, else 1.0. These confidence levels reflect the model's belief that an object exists in that cell and that the bounding box is correct. Intersection over union or IOU is the area of the intersection of the predicted and ground truth boxes divided by the size of the union of the same predicted and ground truth boxes. The confidence value is determined by the IOU. Each cell predicts the object's class in addition to generating bounding boxes and confidence scores. A one-hot vector length C, the number of classes in the dataset, represents this class prediction. It's important to note, though, that while each cell can predict any number of bounding boxes and confidence scores for those boxes, it can only predict one class. So, each grid cell prediction will have the shape C + B * 5, where C is the number of classes and B denotes the number of predicted bounding boxes. Since each box contains (x, y, w, h, confidence), B is multiplied by 5. Because each image contains S x S grid cells, the model's overall forecast is a tensor of shape SxS (C + B*5)[20], [21].

So, each grid cell,

- predicts B boundary boxes, and each box has a box confidence score,
- detects one object only regardless of the number of boxes B,
- predicts C conditional class probabilities.

For example, in figure 4, if an image is divided into 7X7 grids with S =7, 2 boundary boxes B, each with 1 corresponding confidence score and 4 coordinates (width, height, x, y), as well as 4 classes (C), which makes up for tensor.

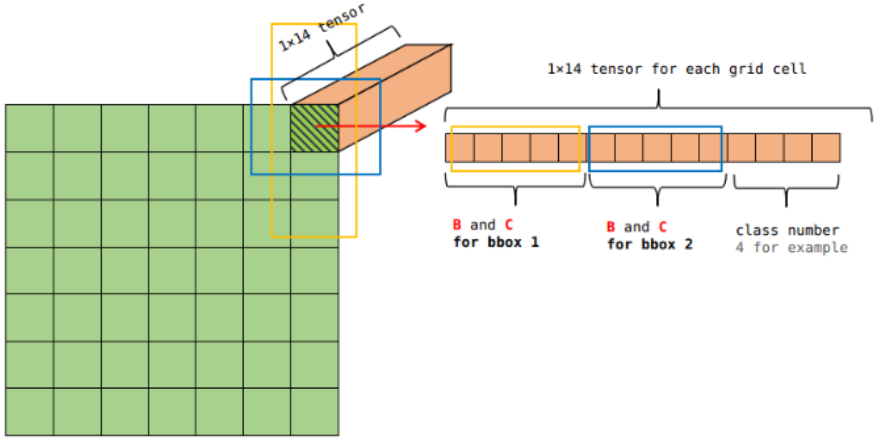

Figure 4 - Grid cell tensors[20]

These values are configured using in a cfg file during the implementation phase of the algorithm. chapter 5 describes the configuration in detail.

YOLO Versions

After the introduction of YOLO in 2016, a series of high-performing object detectors are produced based on the YOLO implementation. YOLO, YOLOv2, YOLOv3, and now, YOLOv4 and YOLOv5 are a few to name. There are other revised-limited versions like YOLO-LITE and YOLOv4-tiny other than the main versions [22]. The latest main version is YOLOv5. This is a Pytorch implementation of the algorithm and is the only main version released without a supporting academic paper and it is still in development[22], [23] and so the latest stable main version is YOLOv4, and it is the version selected for this study.

YOLO Architecture

The head, neck, and backbone are the three main components of the YOLO model. The backbone of the network is made up of convolutional layers that recognize and process key characteristics in an image. The backbone is often trained at a lower resolution than the final detection model, as detection requires finer details than classification. An image database, for computer vision and deep learning research, known as ImageNet[24] is used to train backbone. To create predictions on probabilities and bounding box coordinates, the neck combines features from the backbone's convolution layers with fully connected layers. The head is the network's final output layer, which can be interchanged for transfer learning with other layers with the same input shape.

CSPDarknet53 is a convolutional neural network and backbone for object detection that uses DarkNet-53[25]. It employs a CSPNet strategy[26] to partition the feature map of the base layer into two parts and then merges them through a cross-stage hierarchy. The fundamental goal of CSPNet was to make it possible for this architecture to produce a richer gradient combination while minimizing computation time. This is accomplished by splitting the base layer's feature map into two sections and then merging them using a proposed cross-stage hierarchy[27].

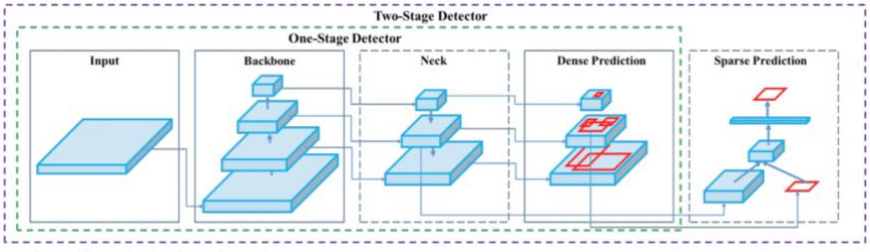

Figure 5 - YOLOv 4 Architecture[28]

YOLOv4 is built on recent research findings, with the backbone being CSPDarknet53, SPP - Spatial Pyramid Pooling [29] and PAN - Path Aggregation Network[30] for the Neck and YOLOv3 for the Head [28].

These three parts of the model work together to extract significant visual aspects from a picture, classify them, and then bind them[20], [21], [31]. Figure 2 shows the steps in object detection by YOLO.

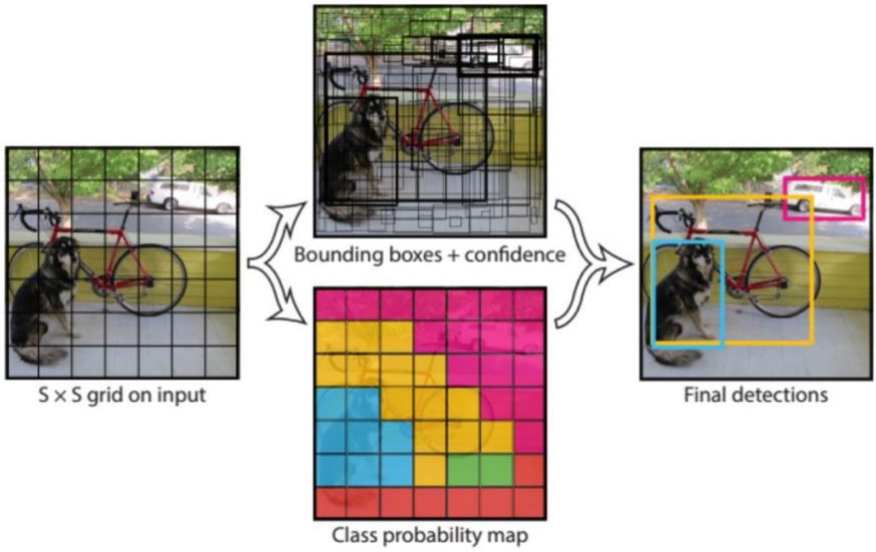

Figure 6 - Object detection steps[21]

YOLO Loss Functions

To calculate loss, YOLO employs the sum-squared error between predictions and ground truth for the ones with the highest IOU[20].

The loss function is made up of the following components:

- the loss of classification
- the loss of localization
- the confidence loss.

4. Method

In this section, different learning methods of the YOLO algorithm are considered for solving the problem. The object detection from images is done with two different approaches. Transfer learning and model training. Transfer learning is when the algorithm is pre-trained, and that knowledge is re-used for solving a similar problem. The other method is the normal model training approach in which the model is trained on individual data and then it is used for detection purposes. This method of training YOLO model using custom dataset is referred as custom data training. YOLO algorithm expects data in a particular format. Along with the input images, the bounding box information of the objects in the images must be set and feed to YOLO as input. There are multiple ways to prepare the data for this. Either the already existing datasets can be used, or the data can be prepared manually. The summary of the training methods used in the study is outlined in figure 7. More details about each method are described in the following sections.

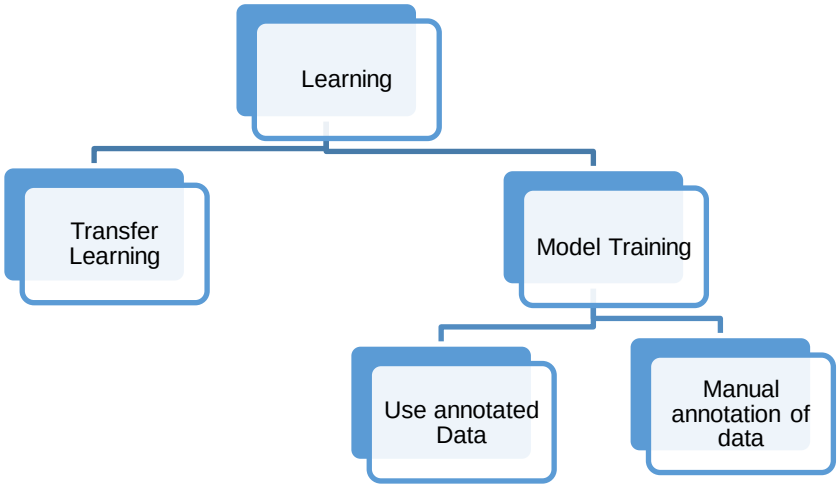

Figure 7 - Different training methods

4.1 Object Detection using transfer learning

YOLO models are pre-trained on a large object detection dataset called MS COCO dataset[32]. The dataset contains 80 categories and some of them belong to the interested classes for this paper. Banana, broccoli, and carrots are some of such categories. More details about this data set are described in chapter 5.1. So as the model is aware of objects belonging to the COCO dataset, for detecting those categories, separate training is not needed.
The categories in COCO that are relevant to this study are apple, orange, banana, carrot, and broccoli. These 5 classes are pre-trained on the YOLO model.

4.2 Object Detection using Model training

For the detection of other categories which do not belong in the pre-trained dataset, custom training is needed. As discussed, YOLO expected data in a particular format. The images and the objects' bounding box information are needed as input for YOLO.

The dataset should have the images and corresponding object details in a text file with the same name as that for the image file. The text file contains the object bounding box information with the class name. All the class names are defined in another file and that is also fed as input to the algorithm. More details are discussed in the implementation, dataset sections in chapter 5.

As discussed above, the training is done with two different datasets. One is google open images [33] which is an annotated images dataset for computer vision. The other one is a manually collected and annotated dataset.
In the study two categories tomato, potato images are considered for custom training. The same categories are used to compare the performance of the algorithm on same set of class. 300 images from both potato and tomato are collected from google open images and 100 images from both categories are manually labelled for the custom training. More dataset related details are discussed in the dataset section of chapter 5.

5. Implementation

Google colab notebook [34] with GPU as Hardware accelerator is used for implementation. Since YOLOv4 is based on darknet framework, as a first step, the darknet framework [18] is cloned from the repository[35] and built by enabling OPENCV - Open Source Computer Vision Library [36] and GPU [37].

Transfer learning

The pre-trained weights for COCO dataset are available in the darknet repository. The configuration file is also available. Using the default configuration file and the weights the detector is run for test marketing flyer images. The COCO dataset classes such as banana, carrot, orange, broccoli and apple are detected with confidence values above 0.7.

Custom training

For creating YOLOv4 detector for custom data classes, the following details are needed:

- Labeled Custom Dataset
- Custom .cfg file for configuration
- obj.data and obj.names files with class and data information
- train.txt and test.txt with object bounding box information in images

As discussed in the methods chapter, the custom dataset is prepared in two different ways. Both methods of data preparation details are mentioned in the

dataset section of the chapter. The prepared data is uploaded to the personal google drive and is mounted with the google colab to avoid any data loss and for access convenience. The data is then moved from google drive to the cloud VM of the colab.

YOLO Configuration

The next step is to configure the model according to the data and our needs. The sample cfg file is copied from the darknet and configured according to our requirements. Configurations are set as per the paper[38]. Width and height are set to 416 chosen as a standard resolution size[39] and the classes are set to the number of classes that needs to be identified.

There are some equations to be followed for the model to work and for the best results. max_batches is set by the number of classes to be identified * 2000. But it cannot be less than 6000 so for identifying classes up to 3, it will be 6000. For more classes follow the equation. The steps will be 80 and 90 percent of max_batches. Filter calculation is as follows: filters = (# of classes + 5) * 3. This is according to the tensor size as described in chapter 2 YOLO section.

Input to YOLO Algorithm

After configuration, the collected annotated data is to be fed to the algorithm. It should also specify the training and test image directories separately.

The model is ready for training now and it will take a couple of hours to complete the epochs. As the colab goes idle during the training, there is a chance for losing the so far trained model and weights and to avoid that after every 100 iterations, a weights file is saved to google drive. It is also possible to start training from the last saved weights file so that a restart is not needed for the training.

5.1 Dataset

There are three sets of datasets used for the experiment. The pre-trained YOLO model used the MS COCO dataset, and the custom training used the google open images dataset and manually collected pictures. All data belong to fruits and vegetable categories as they are the more frequent and repeating products in grocery marketing flyers.

The pre-trained MS COCO dataset has 328K images in 80 different object categories [32]. The categories among them that are of interest for the study are carrot, banana, broccoli, orange, apple, and so on. So, the model weights obtained after training with COCO are directly used for identifying objects from these 5 classes.

The second one is the google open images dataset[33]. Google's Open Images is a massive collection of images. It is one of the most comprehensive datasets in the computer vision community, with over 9 million images, 80 million annotations, and 600 classes spanning numerous tasks.

The third set is the same category images of potato and tomato collected from Kaggle dataset[40] and manually annotated the object positions using labelImg tool[41]. The same categories of data is selected for comparing the detection performance as the model trained with google images did give mean average precision of 46% only.

Data collection

The data collection step is done for the google open images and the manually annotated image dataset.

From the google open images dataset, images from the categories potato and tomato are selected and 300 images from both the classes are downloaded using the OIDv4 toolkit[42]. The number of classes are limited to avoid issues with the storage and GPU utilization limitations of google colab. It took around 2 hours to complete the download of 600 images from both categories and the data is split into 80, 20 percent for training and testing.

For the Kaggle images, the dataset is downloaded and using the labelImg tool, 100 images of each category are annotated with bounding box information of objects. The tool has the option to select the algorithm and then the objects are marked and labelled. Labelling of 200 images took less than an hour to complete. The data is then split into training and test folders in 80:20 percent.

YOLO algorithm expects data in a particular format. For each image in the dataset, the bounding box information of the objects in the image must also need to be set and fed to YOLO as input. The YOLO format of bounding box information is shown in figure 8.

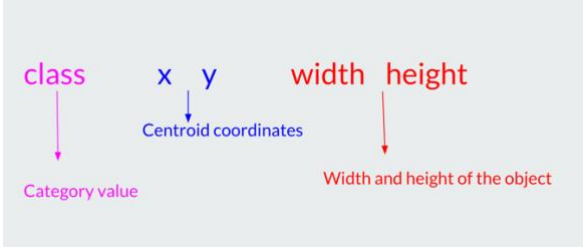

Figure 8 - Bounding box information format for YOLO

Along with the collected images, the bounding box information of the objects belonging to the images also needs to feed to the YOLO algorithm. So, for each image, there will be a corresponding text file with bounding box information of the objects in the image. If there are multiple objects in the image, the object details are to be entered in separate lines in the text file.

The first value is the class name, where the classes are specified in another text file in sequential order and that class value is set in the object text file as the first parameter class. x, y values are the center coordinates of the object. Width and height are the width and height of the object. These values are all derived from

the bounding box information. The images, text file and class file are the input to the algorithm.

The OIDV4 tool and LabelImg tool have scripts to create the bounding box information in YOLO format as described above.

Data Analysis

After collecting the data, analysis is done and some of the images in both google open images and Kaggle do not represent the objects as in marketing flyers. For the manual annotation, only the images that represent the flyer objects are kept for annotating as there is control over the image selection in this method.

6. Results

The trained weights are used for testing detection. There is a flag '-thresh' that can be used to add a threshold for confidences on the detections. Only detections with a confidence level above the threshold set will be returned on detection after training.

The pre-trained model has a 43.5%AP for the MS COCO dataset at a real-time speed of~65 FPS on Tesla V100[38]. The pre-trained model is tested against marketing flyers with the categories in the COCO dataset and the results are shown below.

Figure 9 - Result from pre-trained classes

The custom dataset training was slow and google colab stopped several times before completing the training. The model supports restarting training from the last created weights, and since after each 100th epoch weights were stored in google drive, the training could complete the desired epochs.
The training details for the custom dataset after 1000 epochs are as follows:

Last accuracy mAP@0.50 = 46.47 %, best = 48.07 % with 3.090093 avg loss, 0.001000 rate.

The loss values are as follows:
class_loss = 3.761719, iou_loss = 3.044290, total_loss = 6.806009
total_bbox = 1725753, rewritten_bbox = 0.279472 %

The training details for the custom dataset after 2000 epochs are as follows:

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.433023, or 43.30 %

After 2000 epochs the mAP value is reduced.

Figure 10 - Results from custom trained classes

The manually labeled custom dataset training details are as follows, after 2000 epochs:

Last accuracy mAP@0.50 = 95.51 %, best = 98.56 %
class_loss = 0.000063, iou_loss = 0.318800, total_loss = 0.318863

The loss function of different training are represented in the below figure.

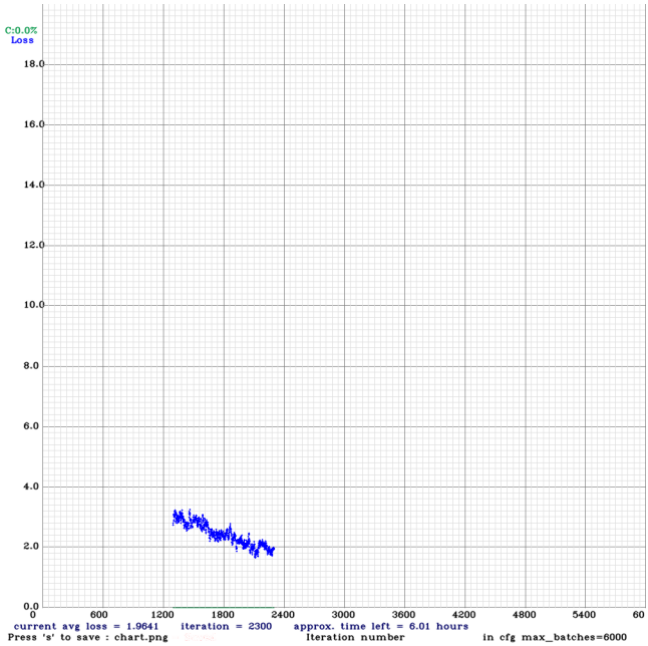

Figure 11 - Loss function of google open images data training

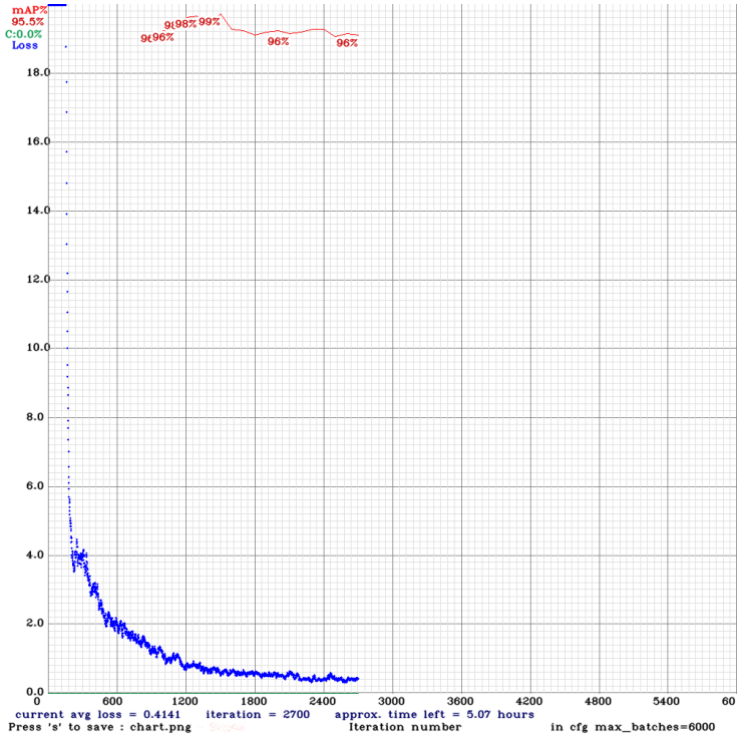

Figure 12 - Loss function of manually annotated dataset trainin

The results of different training are summarized in the following table.

Dataset	mAP	IoU loss	Class loss	Toal loss	Hardware
COCO (pre-training)	43.5%				Tesla V100
Google open images	48.07%	3.04	3.76	6.81	NVIDIA (R) Cuda compiler driver
Manually annotated custom dataset	98.56%	0.32	0.00	0.32	NVIDIA (R) Cuda compiler driver

From the results, the YOLO model trained on the custom dataset is giving 98.56% mAP. So, for building high-performing models, different training approaches with different datasets can be experimented.

7. Discussion

The pre-trained model mAP is 43.5%, but the categories are getting detected from the marketing flyer images without any additional training.
But the prediction confidence is not stable, it is showing 90 or above in some cases and some cases less than 30. The lesser confidence is when the object is partly visible.

The model is identifying individual objects separately, instead of identifying oranges or apples as a single group, it is detecting each item separately. This is because the objects in the COCO dataset are individual. But normally the flyers contain objects like apples, oranges, or carrots as a group and the objects like cabbage, and broccoli as single images.
The custom-trained model is detecting the objects as a group because the dataset contains objects in a group.

So, it is not possible to use a model trained on individual data directly to detect the objects in a group. But it is possible with identifying the nature of objects in flyers and getting the same representative dataset and with custom training of the model using those data.

Another interesting thing found was the detection of other classes like the dining table when there is a combination of multiple items like cake bottles and donuts. This is out of the scope of the paper, but it is giving an insight into detecting more

than one object in a single bounding box which was a limitation of the algorithm[38]. Figure below shows such a result.

Figure 13 - Results from pre-trained classes other than fruits and vegetables

To answer the research questions,

1. Can dataset of individual objects be used for training single-stage object detector like YOLO for detecting objects from marketing flyers containing number of objects?
   Yes, individual objects datasets can be used to train the YOLO algorithm and detect images from marketing flyers. But the data must be the actual representative of the objects appearing in flyers. So a pre-trained model may not work, but a custom trained model will work if the data representation is accurate.
2. How do combine different training methods of YOLO to detect objects from marketing flyers?
   Pre-trained models and custom trained models can be combined to achieve better results. But the training data must represent the actual data in the flyers. In the experiment above the COCO dataset is not representing the actual data in groups for some categories.

7.1 Ethical consideration

As this research does not deal with any personal or sensitive data. However, collecting advertisements may affect marketing negatively, especially the smaller retailers. The price of the products may fluctuate with the advertisements of peer

shops and sometimes retailers may be forced to sell products at a cheaper price than the actual cost of the product by comparing other shops' prices.

Another possible ethical issue is related to advertisements. Advertisements of harmful products with less prices may make the customers buy unnecessary things.

8 Conclusion

Pre-trained models are not sufficient for real-time detect problems. But they can be fine-tuned to train actual representative data to get high accurate models.
If the situation demands, manual data preparation can be done as in this case, manual labeling of images is done. This will give more control over the type of data being fed to the model and more accurate results can be achieved.

8.1 Future Work

To solve the object detection problem completely, the marketing flyers are to be studied deeply and categorize the products and product images. Then collect the best representative data to train and validate the model with the custom training method. This way an accurate model is achieved to detect all product categories from flyers.

To solve the actual problem of content extraction, text detection also needs to be implemented to get the price values of the products. A merging algorithm to map the product and price using the distance is to be implemented as well.

References

[1] L. Jiao *et al.*, "A Survey of Deep Learning-based Object Detection," Jul. 2019, doi: 10.1109/ACCESS.2019.2939201.

[2] G. Boesch, "Object Detection in 2022: The Definitive Guide," *viso.ai*. https://viso.ai/deep-learning/object-detection/ (accessed Jun. 01, 2022).

[3] H. P. Mosquera and Y. Genc, "Recognition and Classifying Sales Flyers Using Semi-Supervised Learning _ Enhanced Reader," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, 2019, pp. 1–6. doi: 10.1109/UBMK.2019.8907146.

[4] N. Cerna, "2018 DIRECT MAIL FACTS & FIGURES," *https://dma.org.uk/*, Feb. 25, 2018. https://dma.org.uk/article/2018-direct-mail-facts-figures (accessed Apr. 01, 2022).

[5] "How Effective Are Flyers for Marketing in 2020," *London Circular distribution*, 2018. https://londoncircular.co.uk/how-effective-are-flyers/ (accessed Apr. 01, 2022).
[6] I. Gallo, A. Zamberletti, and L. Noce, "Content extraction from marketing flyers," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9256, pp. 325–336. doi: 10.1007/978-3-319-23192-1_27.
[7] E. Apostolova, P. Pourashraf, and J. Sack, "Digital Leafleting: Extracting Structured Data from Multimedia Online Flyers." [Online]. Available: http://pdftohtml.
[8] Chen Chao, "On-device Supermarket Product Recognition," *Google AI Blog*, Aug. 11, 2020. https://ai.googleblog.com/2020/07/on-device-supermarket-product.html (accessed Apr. 24, 2022).
[9] A. Calefati, I. Gallo, A. Zamberletti, and Noce Lucia, "Using Convolutional NeuralNetworksfor ContentExtractionfrom Online Flyers".
[10] R. Sarkhel and A. Nandi, "Visual segmentation for information extraction from heterogeneous visually rich documents," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Jun. 2019, pp. 247–262. doi: 10.1145/3299869.3319867.
[11] S. Milyaev and I. Laptev, "Towards reliable object detection in noisy images," *Pattern Recognition and Image Analysis*, vol. 27, no. 4, pp. 713–722, Oct. 2017, doi: 10.1134/S1054661817040149.
[12] François Chollet, "Transfer learning & fine-tuning," *Keras guide*, Apr. 15, 2020. https://keras.io/guides/transfer_learning/ (accessed Jun. 03, 2022).
[13] S. Park, "A guide to Two-stage Object Detection: R-CNN, FPN, Mask R-CNN," *medium.com*, Jul. 28, 2021. https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c (accessed May 30, 2022).
[14] P. Rajeshwari, P. Abhishek, P. Srikanth, and T. Vinod, "Object Detection: An overview," 2019. [Online]. Available: http://creativecommons.org/licenses/by/4.0

[15] L. Tan, T. Huangfu, and L. Wu, “Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identiication,” 2021, doi: 10.21203/rs.3.rs-668895/v1.
[16] “Difference between YOLO and SSD,” *https://www.geeksforgeeks.org/*, Jul. 18, 2021. https://www.geeksforgeeks.org/difference-between-yolo-and-ssd/ (accessed Jun. 02, 2022).
[17] J. Nelson, “Your Comprehensive Guide to the YOLO Family of Models,” *blog.roboflow.com*, Jun. 07, 2021. https://blog.roboflow.com/guide-to-yolo-models/ (accessed May 16, 2022).
[18] J. Redmon, “Darknet: Open Source Neural Networks in C,” *pjreddie.com*, 2016. https://pjreddie.com/darknet/ (accessed May 31, 2022).
[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection.” [Online]. Available: http://pjreddie.com/yolo/
[20] Handuo, “You only look once (YOLO) -- (1),” *github.io*, Aug. 20, 2018. https://zhanghanduo.github.io/post/yolo1/ (accessed May 16, 2022).
[21] A. Aggarwal, “YOLO Explained,” *medium.com*, Dec. 27, 2020. https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31 (accessed May 16, 2022).
[22] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, “A Review of Yolo Algorithm Developments,” in *Procedia Computer Science*, 2021, vol. 199, pp. 1066–1073. doi: 10.1016/j.procs.2022.01.135.
[23] J. Solawetz, “YOLOv5 New Version - Improvements And Evaluation,” *https://blog.roboflow.com*, Jun. 29, 2020. https://blog.roboflow.com/yolov5-improvements-and-evaluation/ (accessed Jun. 03, 2022).
[24] “ImageNet,” *https://www.image-net.org/*, Mar. 11, 2021. https://www.image-net.org/ (accessed Jun. 03, 2022).
[25] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement.” [Online]. Available: https://pjreddie.com/yolo/.
[26] S.-H. Tsang, “Review — CSPNet: A New Backbone That Can Enhance Learning Capability of CNN,” *medium.com*, Aug. 21, 2021. https://sh-tsang.medium.com/review-cspnet-a-new-backbone-that-can-enhance-learning-capability-of-cnn-

da7ca51524bf#:~:text=Cross%20Stage%20Partial%20Network%20(CSPNet,as%20DenseNet%2C%20ResNeXt%20and%20ResNet. (accessed Jun. 03, 2022).
[27] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.11929
[28] D. Cochard, "YOLOv4 : A Machine Learning Model to Detect the Position and Type of an Object," *https://medium.com/*, Dec. 09, 2020. https://medium.com/axinc-ai/yolov4-a-machine-learning-model-to-detect-the-position-and-type-of-an-object-4f108ed0507b#:~:text=The%20architecture%20of%20YOLOv4,YOLOv3%20for%20%E2%80%9Cthe%20Head%E2%80%9D. (accessed Jun. 03, 2022).
[29] K. He, X. Zhang, S. Ren, and J. Sun, "LNCS 8691 - Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," 2014. [Online]. Available: http://arxiv.org/abs/1406.4729v1.
[30] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation." [Online]. Available: https://github.com/
[31] Y. Chen, M. C. Goorden, F. J. Beekman, and J. Du, "Understanding of Object Detection Based on CNN Family and YOLO You may also like Convolutional neural network based attenuation correction for 123 I-FP-CIT SPECT with focused striatum imaging Understanding of Object Detection Based on CNN Family and YOLO," *J. Phys*, p. 12029, 2018, doi: 10.1088/1742-6596/1004/1/012029.
[32] T.-Y. Lin *et al.*, "LNCS 8693 - Microsoft COCO: Common Objects in Context," 2014.
[33] "Open Images Dataset V6," *google*. https://storage.googleapis.com/openimages/web/download.html (accessed Mar. 30, 2022).
[34] "Welcome to Colaboratory." google. Accessed: May 16, 2022. [Online]. Available: https://colab.research.google.com/?utm_source=scs-index
[35] AlexeyAB, "AlexeyAB/darknet: YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet)," *github*, Oct. 30, 2021.

https://github.com/AlexeyAB/darknet (accessed May 16, 2022).
[36] "OpenCV," *https://opencv.org*. https://opencv.org/ (accessed Jun. 03, 2022).
[37] "Cloud GPUs," *https://cloud.google.com/*. https://cloud.google.com/gpu (accessed Jun. 03, 2022).
[38] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection." [Online]. Available: https://github.com/AlexeyAB/darknet.
[39] J. Solawetz, "YOLOv4 - Ten Tactics to Build a Better Model," *https://blog.roboflow.com/*, Nov. 13, 2020. https://blog.roboflow.com/yolov4-tactics/ (accessed Jun. 03, 2022).
[40] K. Seth, "Fruits and Vegetables Image Recognition Dataset," *kaggle.com*, 2021. https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition (accessed Mar. 30, 2022).
[41] "LabelImg." github. Accessed: May 16, 2022. [Online]. Available: https://github.com/tzutalin/labelImg#macOS
[42] "OIDV4 Toolkit." github. Accessed: May 16, 2022. [Online]. Available: https://github.com/theAIGuysCode/OIDv4_ToolKit