Automated Retail Shelf analysis

Glossary

Planogram Compliance	this refers to the process of making sure that the planogram meets the requirements specified in the diagram
FMCG	Fast-Moving Consumer Goods, they are **products that are sold quickly and at a relatively low cost**
Facings(retail)	Number of products visible on a shelf facing the customer
Functional requirements	technical requirements that explain the low level design objectives
Non functional requirements	requirements ensuring the system's usability, reliability, and efficiency

Objective

Automate retail shelf analysis using images to enable FMCG companies to perform an inspection regarding the guidelines for the number of shelf facings of their products on retail stores selling their products remotely without human intervention.

The Problem

Retail stores receive a price proportional to the number of facings of a particular product on a shelf as well as number of such facings that are at eye level and a few other factors from FCMG companies that supply their products. This project attempts to solve the problem of finding number of facings and percentage of a shelf occupied by a certain brand from images of shelves.

Current process

The current process is to send an inspector to retail stores at regular intervals to ensure that the guidelines are being followed.

Limitations:

- large manual labour is involved in the task of inspection
- frequent inspection is not possible, hence this method is unreliable and easily circumventable.

Customers

The customers for this solution are:

- FMCG companies that adopt this method of inspection
- Retail stores that agree to be inspected by this model

Functional requirements:

- Identify products from a shelf
- classify the products as being of a particular brand or not

Non functional requirements:

- Scale:
  - The proposed phase 1 solution(which is currently implemented) requires retraining of the classifier on new labelled data to use the solution for a new brand which will take around 6 hours of work to create the new dataset and train on it
  - The proposed phase 2 solution requires no additional training to generalize to a new brand.
- Analysis time: Currently takes 8 seconds per image without parallelism.
- ML Accuracy:
  - mAP50 of 0.872 for object detection

    - 83.49% accuracy on the classifier on the validation dataset.
- Cost: For deployment on a cloud server, Rs 2000 is the approximate base cost per month and an additional Rs 50 per month for each retail store to be checked
- Latency: 1 minute per retail store on the server

Out of Scope

- classifying products that are hidden behind others(not counted as a facing)
- counting the number of front facing products on the shelf:
- classifying non front facing products by brand
- analyzing products not placed on shelves
- Setting up a UI to take and upload images of shelves which checks if the angle between the line of sight and the perpendicular to the shelf is not greater than 45 degrees

Assumptions

- The angle between the line of sight of the camera and the perpendicular to the shelf is less than 45 degrees
- The brand maintains its design pattern while making new products

Additional Problems to solve

- Collecting a dataset for classification
- Due to the nature of the dataset, it will be unbalanced, countermeasures should be taken to make the model unbiased despite this.

Technology Considerations

1. Server having storage of atleast 1 MB per retail store being considered and 100 MB for python packages
2. python3 installed and storage available for packages in requirements.txt

Approaches to the main problem:

1. Single model using YOLO[1]:

- In this solution, the YOLOv8 algorithm would be used to identify and classify the products in one step

Pros:

- YOLO is a state of the art model for object detection and classification and would provide a model with minimal error
- A single model solution would be lightweight and use less resources to run

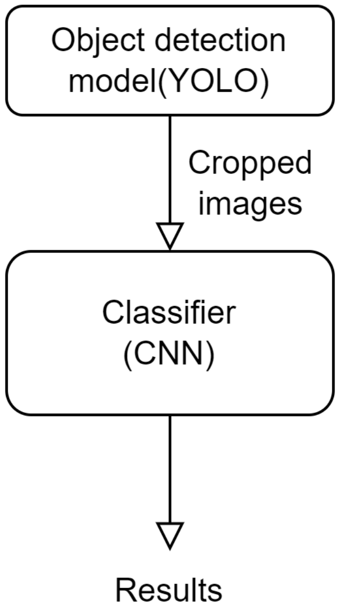
Cons:

- The data required for training such a model which is specific to the application must be created manually
- The data is complex and it is infeasible to create a comprehensive dataset of such data manually for the application
- If created, the dataset would be specific to a particular brand and cannot generalize to other brands
- Further, the model will not be robust to changes such as new product releases as the same data collection process must be repeated each time and the model must be retrained

2. Two phase model with YOLO and a CNN(Recommended phase 1, currently implemented):

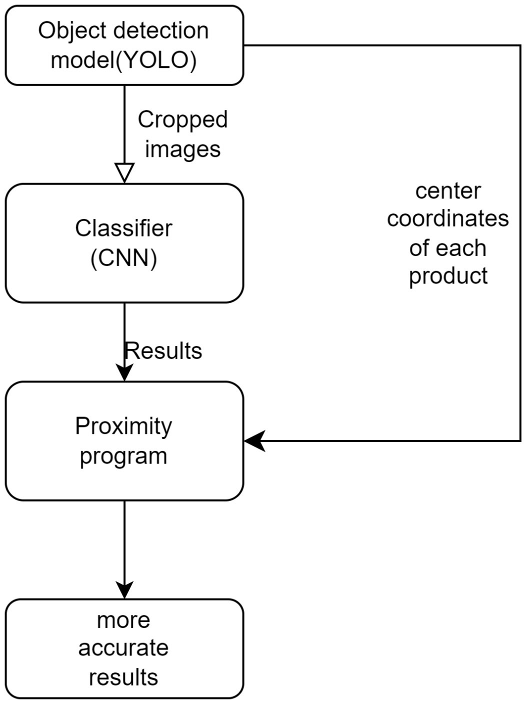
- This model involves detecting objects using YOLO, the classification is done by a separate CNN after cropping out the products from the image using results from YOLO.

A simple diagram that depicts the model is as shown below



Pros:

- By dividing the problem into independent parts, we allow for further improvements to the solution without making a new solution from scratch. One such improvement is to use proximity data from the image i.e. products surrounded by many similar products are more likely to be of the same type. This can be used to classify ambiguously classified products. The diagram below describes such an arrangement



- The data to be collected is of a simpler format:

Cons:

- The solution is still not as flexible to updates as it can be, the CNN still needs to be retrained after major product releases to be up to date.
- The dataset for the CNN is still brand specific and has to be collected again for each brand separately

3. Two phase model with YOLO and a Siamese network[3] (recommended phase 2)

- This solution has a high initial cost to collect a dataset and hence could not be implemented, however this solution is the most robust and scalable approach which can generalize to all brands without retraining any model
- The structure of the solution is as follows:
    - YOLO is used exactly as in the previous approach
    - A database of products belonging to the brand is maintained(2-3 images per product), if there are new product releases, a few images of the

    - product are added to the database
    - A siamese network(trained on Triplet Loss) is run on the cropped images to check for matches with the database and if there is a match, the product is classified as being of that particular brand else it is not

Pros:

- The scalability of the solution is much greater than the other 2, this solution can generalize to any brand without ever retraining the model after the first time just by changing out the database of reference images.
- New product releases can easily be adapted into the model just by adding 2-3 images to the database
- Further, this still holds the advantage of a modular solution which allows for further additions like the proximity program suggested in the previous approach and makes the solution easier to expand to a variety of use cases and easier to improve.

Cons:

- The initial cost is exorbitant as it requires a handmade dataset of at least 10,000 images to get a good accuracy
- This solution is more sensitive to orientation of the products, i.e. products turned more than 30°

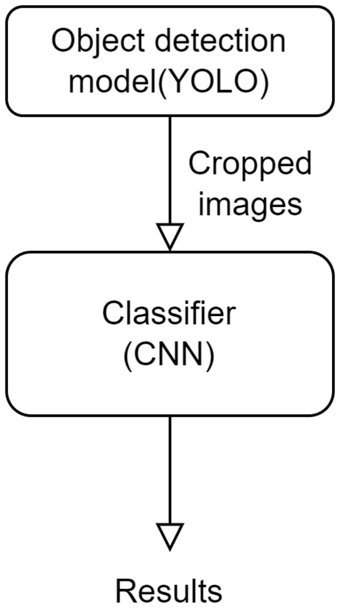
Note: The sensitivity of the model may not be an issue depending on planogram requirements. It may be advantageous if the client requires us not to count products having a tilt greater than 30° in the facings.

Chosen Architecture and Design

The current pipeline is as follows

- Identifying objects was done by training a model using the YOLOv8 algorithm on the SKU110k dataset

- Classification of objects by brand was done by a simple CNN by fine tuning ResNet50 (a pre-trained model trained on ImageNet) on our handmade dataset



- A jupyter notebook was made which describes how proximity analysis can be done in the code base as proximity.ipynb

The Additional problems:

Creating a Dataset

- To train the CNN we made a dataset of products from images taken from a retail store
- The images are classified as 'himalaya' and 'not himalaya' and the dataset is in 2 csv files with an associated folder of images
- The csv files contain 2 columns image path and label
- The dataset is available on kaggle at https://www.kaggle.com/datasets/sohanthummala/classify-by-brand-dataset

Problems with the dataset

- The dataset is skewed, there are far more images of ‘not himalaya’ than ‘himalaya’
    - This will always be the problem for any such dataset that separates from brand from all the others because the number of products of each other brand should also be similar to the brand being identified, causing the negative set to be larger

The solution

- Oversample the minority class to get a more balanced dataset

The other option is to undersample the majority but that will lead to a reduction in the size of the dataset and gives worse accuracy

FAQs

1. Q. In what way is the data simpler to collect in approach 2 as compared to approach 1?

    A. In approach 1:

    - The images need to be labelled with bounding boxes and each bounding box needs to be classified based on the product within it

    In approach 2:

    - For the YOLO model, the dataset need not be tailored to the application as it only needs to be able to identify products, not specific to the brands that we need. The SKU110k dataset[2] can be used for this purpose.
    - For the CNN, the data just needs to be labelled as himalaya and not himalaya, a simple script was written that displays an image and on a single key press labels the image as himalaya or not himalaya

2. Q. What is the structure of the database mentioned in approach 3?

A. There is no limitation on the form or structure of the database, any database that allows for iteratively accessing every image can be used

3. Q. Why was the problem of counting number of front facing products on the shelf declared out of scope?

   A. This was a decision made based on the following:

   - The problem of counting number of front facing products was initially proposed because classifying non front facing products by brand is difficult
   - This problem is of lower priority as products that are not front facing will not count as a facing(based on the definition of facing) and will be classified into the negative class which has a negligible effect on the result(some non front facing products may still be correctly classified into the positive class so there is a small effect)

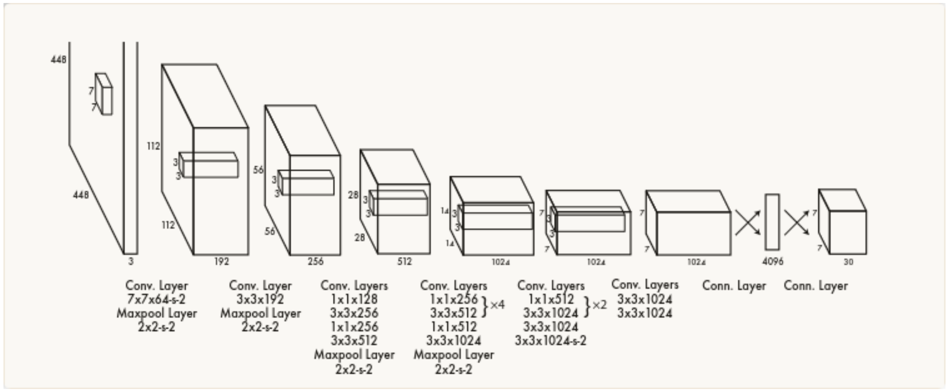
4. Q. How can the Siamese network in approach 3 be trained?

   A. The following is a description of the dataset as well as the steps involved in training a model:

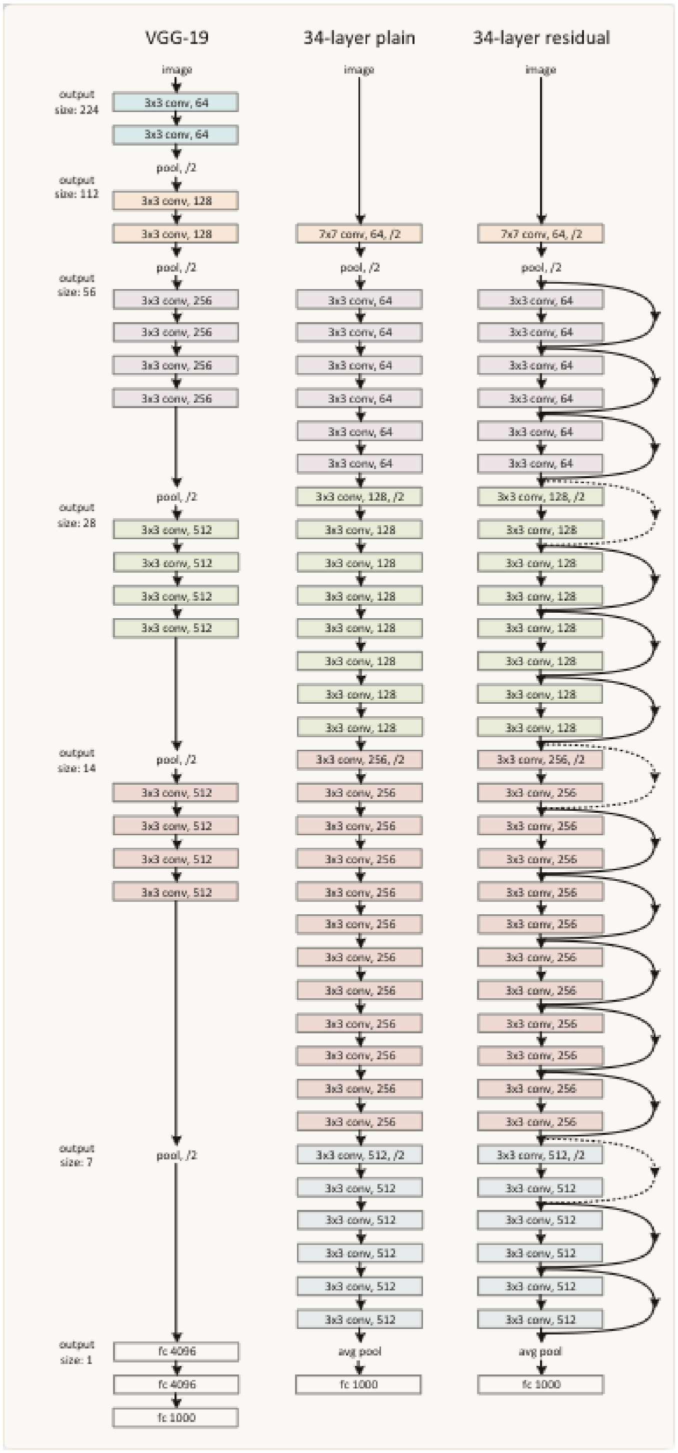
   - Siamese networks are normally used for face recognition security systems, there is a database of faces of people to be allowed in, if the network finds that a person with a face that matches one of these faces tries to enter, they are allowed else they are rejected.
   - Get a list of products(around 50) of various brands, these are hereon referred to as our classes
   - For each class, get around 500-1000 images
   - Choose a class at random and pick an image, this is the image to be recognised
   - Now, pick an image from the same class called the positive example and another image from another random class that is different from the first called the negative example and pass it to the model
   - Train the model using Triplet Loss[4] which takes 3 vectors as input

Annexures

1. YOLO algorithm:
    - This is the current state of the art algorithm for object detection tasks and was the best choice for obtaining bounding boxes
    - The architecture is as shown in the diagram below taken from [1]



2. ResNet50:
    - It is a model that won 1st place on ImageNet detection 2015 and is commonly used as a feature extractor
    - It is different from a simple CNN in the sense that it implements *identity mapping*[5] which can be understood by the comparative diagram below(taken from [5]) that compares another powerful neural network that carries out image classification tasks



References:

1. https://doi.org/10.48550/arXiv.1506.02640 Joseph Redmon et al YOLO algorithm

2. https://doi.org/10.48550/arXiv.1904.00853 Goldman et al SKU110k dataset
3. https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf Gregory Koch et al Siamese networks
4. https://doi.org/10.48550/arXiv.1412.6622 Elad Hoffer et al Triplet Loss
5. https://doi.org/10.1109/CVPR.2016.90 Kiaming He et al Deep Residual Learning(ResNet)