# Towards Automating Price Attribution for Beverage Products on Retail Displays

Stockton Jenkins
jsjenkins4@wisc.edu

Salsabil Arabi
arabi2@wisc.edu

Tyler Thompson
tnthompson5@wisc.edu

## 1. Introduction

### 1.1. Background and Motivation

Consumer product good (CPG) distributors incur a tremendous cost in terms of time, labor, and capital to deliver product to consumers in retail stores. This cost includes paying for shelf space and optimizing pricing in thousands of stores. Field employees are expected to stock displays according to plans given by sales officials. Displays should match a "ground truth" product set, with the correct price for each product. Poor execution can reduce the profit margins of these distributors, making it a high priority for management.

The market varies in many dimensions across stores, making both the product mix and price sets non-constant. This variation make it infeasible to manually measure execution at scale with confidence. Automating this process would reduce the cost of measuring execution compliance and provide valuable insight to the key decision makers in a CPG corporation.

Due to the recent progress in deep learning, computer vision shows promise as an avenue for automation. In this project, we work towards an automated pipeline in solving this problem. The three sub-components we developed over the course of this semester are:

***Box Detection*** We first identify the bounding box coordinates of of both product facings and price tags with two object detection models. Both of which are `YOLO` [8]. We fine-tune the most recent model from the Ultralytics repository (`YOLOv11`, as of this writing) on human-labeled images of beverages and price tags on store displays. We use separate models because of the large discrepancy in the class distribution between any single beverage class and the price tag class.

***Price Extraction*** Multimodal large language models (MLLMs) have been useful in other text-visual problems [7]. We experiment with `LLaVAv1.6` [4] [5] and the free-tier API from Segmind to perform optical character recognition, given a cropped image of a price tag bounding box and a text prompt.

***Graph Construction And Association*** We model the relationship between the components of a display as a graph and learn this representation with a homogeneous graph neural network (GNN). Previous literature has indicated that deep learning can be used to learn the edges of a graph [1]. In our case, the connections (associations) of products and prices can be represented as edges between nodes in a graph. Since no graph labeling data for this problem was previously available, we also manually label the association set between products and price tags in more than 50 images.

### 1.2. Definition

For a display $D$, there exists a product facing set, $P$, and a price tag set, $L$, such that $P = \{p_0, \ldots, p_j\}$ and $L = \{l_0, \ldots, l_k\}$. The products in $P$ have one-to-one relationship to a UPC set, $U = \{u_0, \ldots, u_l\}$. Facings on $D$ can have the same $u$, but not all $u$ are found on $D$. We can represent the set of product-price associations as

$$A = \{(p, l) \mid p \in P, l \in L\}$$

We will denote $A'$ as the ideal (or correct) association set. We aim to construct $A$ such that its IOU with $A'$ is **1**.

The association between $P$ and $L$ is not necessarily one-to-one. For example, price tag $l$ can be the correct price for multiple $p$. These facings can have the same $u$, or the same category (i.e. Sparking/Energy) or brand (i.e. Coca-Cola/Monster). It is also possible for some $p$ to have no correct price tag, especially if it is a newly placed product and the price tag labels have not been updated. The opposite is also true: the facing set for label $l$ might be empty in cases where the product is out of stock (OOS).

## 2. Related Work

To the best of our knowledge, automating price attribution in the manner that we have proposed is not a well-studied problem. However, previous literature applies deep learning to a variety of tasks related to inventory management in retail stores, including real-time inventory visibility, facing counting, and forecasting [2] [3]. Other work combines drones with computer vision to improve efficiencies in warehouse environments [6].

Several previous works also discuss graph neural networks and their applications both in and out of the vision space. Some examples include text classification [13], complex physics simulation [9], 3D object detection [11] [12]. Finally, other work study the use of GNNs in edge prediction between nodes [1] [10], which is directly applicable for our problem.

## 3. Method

In this section, we discuss the three sub-problems previously outlined, as well as the approach we took for each. We also provide details on the image dataset we used for

Figure 1. Detection predictions on test examples for both price tag and product `YOLOv11x` models.

training our `YOLO` models that was provided by Delicious AI [1].

### 3.1. Dataset

We downloaded over $6,000$ display images ($3,000$ unique UPCs) with label annotations from Delicious AI's cloud storage. The dataset had a large class imbalance between the price tag class and any single product class, with about $120,000$ price tag instances and $190,000$ *total* product instances.

Many of the UPCs in the data had few box instances. We pruned the box set to exclude classes with under $k$ instances. We let $k = 100$ in this project. This left $420$ product classes and the price tag class.

### 3.2. Box Detection

The first task in our pipeline is to perform object detection for all $p \in P$ and $l \in L$ for image $i$. We leverage open source software and modeling from the Ultralytics repository. The models from this repository are constructed by `YOLO` [8], a simplified detection algorithm that does not generate region proposals and only requires one forward pass through the network. We used `YOLOv11X` for our experiments.

We divided this subproblem into two; detection on price tags and products was performed independently with two different `YOLOv11X` models. As outlined in 3.1, the class imbalance between all price tags and any single class of products was significant. Splitting products and price tags into two groups and performing inference independently was a trivial solution to this class imbalance. Each `YOLOv11X` model was pretrained by Ultralytics, and we fine-tuned each on its respective box set.

### 3.3. Price Extraction

Given bounding boxes of price tags, we can attempt to extract the price text. These bounding boxes are used to generate crops of each price tag in an image. We used `LLaVA` [4] [5] to perform optical character recognition (OCR). Due to lack of availability to sufficient compute and data, we did not fine-tune LLaVA for our use case. Instead, we leveraged an open-source API provided by Segmind. This API accepts visual inputs (images) and text instructions (prompts) and returns a text response from a `LLaVA` model. Segmind exposed `LLaVAv1.6` with 7B parameters in their endpoint.

Delicious AI did not have pricing label data in the dataset they provided. As a result, we manually labeled the pricing for about 110 cropped images for our experiments.

### 3.4. Box Association

#### 3.4.1 Homogeneous Graph Neural Network

Finally, we associate product boxes and price tag boxes in an image. This problem can be formally represented as a graph, where the set of nodes are bounding boxes, regardless of class, and the relationship (or associations) between them represent the set of edges. We took this approach because previous literature has indicated that a GNN can be used to learn the edges of a graph [1].

We implemented a homogeneous GNN using the `torch geometric` open-source library. Homogeneous graphs have nodes and edges of a single type, meaning all nodes represent instances of the same entity, and all edges represent relations of the same type. While this does not model our problem perfectly, we felt it was sufficient and that it was more straight forward to implement than a *heterogeneous* GNN. This is certainly a direction of exploration and future work.

The input to the GNN includes the embeddings of each node, an adjacency matrix for the graph being predicted on, and source/destination pairs that we hope to predict links between. The node embeddings simply consist of the center $(x, y)$ coordinate of the box, the box's width and height, as well as a binary flag indicating that the box is either product (1) or price tag (0).

The adjacency matrix enables the sharing of features between selected nodes and aggregates information between them. Doing this over several layers eventually propagates information throughout the entire graph.

The model consists of an encoder and decoder. We encode the node embeddings and adjacency matrix with sev-

[1] Delicious AI is a software company based out of Lehi, UT that specializes in computer vision applications for CPG distributors.

eral layers of transformer convolutions. The decoder is comprised of several linear layers, with a tail-end sigmoid to map the logits to edge link probabilities.

We train on both positive and negative examples, as to prevent the model from learning that every edge it is prompted has 100% probability of association. Here, we define the negative examples as the connections that are *plausible*, but don't exist in the data. Under this paradigm, the model must not only learn which connections are real, but also which are "fake".

Figure 2. Processed image with its (green) box annotations for both products and price tags. The number centered in each box was its index in the annotation `.txt` file. Shuffling the order of the annotations was performed pre-labeling.

#### 3.4.2 Labeling

Again, Delicious AI did not have association labels that we could use for training. As such, we manually crafted labels. Due to the tedious and time-consuming nature of the labeling, we only annotated 53 images. First, we joined all the box annotations for each image into a single file and shuffled the order to separate spatially related boxes. Then, we projected each box onto the image with its corresponding index in the annotation list. See figure 2 for an example. With these reference images, we were able to manually denote associations in `.json` files. The keys were the product indices, and the values were list of associated price tag indices.

With the annotations in hand, we wrote code to generate several tensors used for training in each scene. This includes adjacency matrices for the `product-price` connections, `product-product` groups, and `price-price` groups. The `product-price` connections are generated directly from the labeled data, while `product-product` and `price-price` groups are connected components within the graph. Another advantage of this approach is that we can retrieve the connected groups of prices and connected groups of products. This is explained further in section 4.3.

## 4. Experiments and Results

In this section, we detail the experiments we ran for each sub-problem outlined in section 3, along with their results.

### 4.1. Box Detection

As described in 3.2, the dataset with which we trained two `YOLOv11x` models was filtered to only include classes with at least $k$ boxes. This left $4,345$ images in the dataset, with $92,416$ product box instances and $118,288$ price tag instances.

We trained each model for 200 epochs with default `YOLO` training arguments. This includes an Adam optimizer and data augmentations, like hue, translation, mosaic, scale, etc. Each model had over 57 million parameters. Training was done on 4 **NVIDIA GeForce RTX 3070** GPUs with 8GB ram each.

Results are summarized in table 1. Both models achieved above 80% mAP@50. Loss curves were smooth and stable during training and validation. Figure 1 shows an example of both the product and price tag detectors' output on the same pair of images.

### 4.2. Price Extraction

We manually labeled 110 price tags and used `LLaVA` API to extract associated prices. We removed extremely noisy and low-resolution images from consideration. The model was able to extract accurate price tag texts for approximately 69% input images.

### 4.3. Box Association

We experimented with and trained our GNN independently of `YOLO` so that we could maximize the quality of annotations to train on. The GNN was trained for 200 epochs with a positive and negative component of binary cross entropy loss. Edge link probabilities were rounded to the nearest integer (0 or 1).

The final accuracy was computed from the true positive, false positive, true negative, and false negative components of the model output. Our GNN achieved a validation accuracy best of $92.48\%$, with a precision of $89.70\%$ and recall of $97.43\%$. With a much higher recall than precision, we can infer the model is still overconfident in the number of edge links in the graphs it is predicting.

To visualize the output probabilities of our GNN, we wrote code in a Python Jupyter notebook. A sample is shown in figure 3, which shows the model's edge link probabilities for the scene in figure 2.

| Type | Box Loss | Class Loss | Precision | Recall | mAP@50 | mAP@50-95 |
|---|---|---|---|---|---|---|
| Product | 0.322 | 0.190 | 0.784 | 0.836 | 0.859 | 0.764 |
| Price Tag | 0.861 | 0.385 | 0.833 | 0.736 | 0.824 | 0.510 |

Table 1. Detection results for both `YOLOv11x` models (products and price tags).

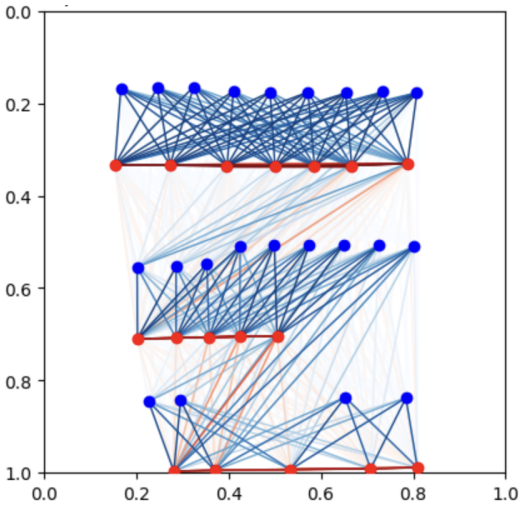


Figure 3. Visual representation of our GNN's edge link probability output for the scene in figure 2. Blue dots and red dots represent product and price tag nodes, respectively. The intensity of the lines between nodes signify the model's confidence in the edge link.

The blue dots and red dots represent product and price tag nodes, respectively. The intensity of the lines between nodes signify the model's confidence in the edge link. Here, we see that the model is learning meaningful structure within the data, and has a great feel for which price tags are grouped together, which products are grouped together, and which products belong with which price tag. We are encouraged by these results, and believe that more training samples would yield higher accuracy and even further delineation between components in the graph.

# 5. Discussion

## 5.1. Future Work

Despite our promising results, we were unable to implement a full pipeline of that connected the components of box detection, text extraction and box association. This was due to two main factors. First, we did not have a labeled dataset of ground truth product-price associations. Constructing a large dataset manually would have taken a significant amount of time. We instead focused on solving each sub-problem. Second, the Segmind API free-tier only allowed 5 requests per minute, thus limiting its use for testing on large sets of images in a short amount of time.

## 5.2. Limitations

There are several limitations to our proposal and lingering challenges to solving this problem. First, a three-stage process inherently contains several failure points. Any errors or mis-classifications in an early step would be propagated and magnified downstream. Second, many price tag crops generated from price tag bounding boxes are blurry and/or noisy. The lack of clearly visible price tags made it more difficult for an off-the-shelf MLLM to perform the OCR task. Perhaps, fine-tuning on a dedicated price tag dataset would improve performance. Third, to the best of our knowledge, labeled graph data for price tag and product boxes is not common, and as such, a large dataset would need to be constructed in order to improve our GNN's performance. Not all graph outputs were as *separable* as figure 3. There were also several examples where out-of-focus price tags detected from neighboring displays were incorrectly associated with products of interest.

## 5.3. Conclusion

In this project, we worked towards a solution to the problem of product-price association of beverage products on retail displays. We broke down the problem into three subproblems: box detection, price extraction, and box association. Although we were unsuccessful in constructing a full pipeline, there is a clear path for future work to put together the pieces we have proposed in this project.

We discovered that the price association problem can be redesigned as a graph edge prediction problem. The nodes of this graph can be represented by bounding boxes predicted by one or more deep models, like `YOLO`. A GNN can then be used to predict edge links between boxes, thus associating them. Price text can then be extracted by an MLLM from the price tag boxes to give a final set of prices for each product facing.

## 5.4. Contributions

All team members contributed to the project and had input and collaborated on each stage. Each team member led the effort on a stage — Tyler led *Box Detection*, Salsabil led *Price Extraction*, and Stockton led *Box Association*.

## References


[1] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018. 1, 2

[2] Porter Jenkins, Kyle Armstrong, Stephen Nelson, Siddhesh Gotad, J. Stockton Jenkins, Wade Wilkey, and Tanner Watts. Countnet3d: A 3d computer vision approach to infer counts of occluded objects. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3007–3016, 2023. 1

[3] Porter Jenkins, Michael Selander, J. Stockton Jenkins, Andrew Merrill, and Kyle Armstrong. Personalized product assortment with real-time 3d perception and bayesian payoff estimation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 5161–5171. ACM, 2024. 1

[4] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. 1, 2

[5] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2024. 1, 2

[6] Chommaphat Malang, Phasit Charoenkwan, and Ratapol Wudhikarn. Implementation and critical factors of unmanned aerial vehicle (uav) in warehouse management: A systematic literature review. *Drones*, 7(2), 2023. 1

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 1, 2

[9] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020. 1

[10] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks, 2020. 1

[11] Weijing Shi, Ragunathan, and Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud, 2020. 1

[12] Yue Wang and Justin Solomon. Object dgcnn: 3d object detection using dynamic graphs, 2021. 1

[13] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification, 2018. 1