6.869 Final Project Human Action Detection

Lauren Heintz
Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge MA
lheintz@mit.edu

Max Tanski
Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge MA
mtanski@mit.edu

Abstract

To understand a visual world, machines must be able to identify objects and determine the way these objects interact with each other. However, these interactions are not trivial and at times can be quite complex. In this paper we explore the dependent relationships between these objects in order to allow a machine to be able to recognize and classify the objects in an image that are acting and the objects that are the targets of these actions. Quite often, humans are at the center of such interactions. Detecting these human-object interactions is an important practical and scientific problem that has application in many domains. We hypothesize that the instance segmentation boundaries of current object detectors can be leveraged to identify both people and their actions is a powerful cue for localizing the objects they are interacting with. We propose an implementation to exploit this cue by creating a model that learns to predict a target object given assumed action based on the instance segmented pose of a detected person.

1. Introduction

The impetus for this topic originates from the popularity of online cookies used on websites to track users' spending and browsing habits. This is common practice in order to gather data on consumer preferences such as: how long they look at a product, whether they click on it to read more information, and whether or not they ultimately buy the product. However, this data is absent for those who do not shop online due to the prohibitive cost of tracking users in a store at scale. Given this large scale and expensive problem, this project focuses on implementing a solution that can effectively track users in physical retail stores through computer vision.

An industry that is especially lagging in online sales is grocery shopping. We speculate that, given the amount of image data of grocery stores that is available, that enough relevant observations could be gathered from video or image data to better understand shoppers' habits and identify how long they spend in an aisle, whether they pick an item up, put it back, or put it in their cart. We further propose that a project like this could yield valuable information for retailers from their brick-and-mortar stores, making them more modern and data-focused.

In order to provide brick-and-mortar grocery stores with this data, we narrow our scope to identifying < human, action, target > triples based on still images of people shopping in a grocery store. We identified a few human-object interactions that would be relevant to a grocery store and focused on just those actions in our models. Examples of human-object interactions we will categorize are: "hold", "carry", "point", "eat", "drink", "stand", "talk_on_phone". While this behavior tracking could have applications in many brick-and-mortar retail settings, we also narrowed our scope to just grocery stores and focused on food related data.

2. Related Works

Human object interaction requires piecing together several different deep learning models and combining them to link together a human (subject), action (verb), and object (target) as a triplet. One aspect of this is object detection in order to identify all the objects in the photo which could ultimately be the human subjects and the target objects. We reference the work done in InteractNet[1] by FacebookAI Research (FAIR) for our object detection method. Another important piece of implementing this pipeline is action detection. Our method was inspired by existing model architecture from both FAIR and the Human Activity Knowledge Engine (HAKE) for our preliminary research. We also leverage an existing labeled action classification databases V-COCO[2] to train and evaluate our model pipelines.

2.1. InteractNet

FAIR (Facebook Artificial Intelligence Research) presented a human-centric model for recognizing human-object interaction that proposed that a person's appearance[1], which reveals their action and pose, is highly informative for inferring where the target object of the interaction may be located. The search space for the target object can thus be narrowed by conditioning on this estimation. Although there are often many objects detected in a scene, the inferred target location can help the model to quickly pick the correct object associated with a specific action.

FAIR codified this intuition as three branches: an object-centric branch to identify objects in a scene, a human-centric branch to identify an agent's action and target, and a final interaction branch that collated the two other branches to produce an HOI triple. For the human-centric branch specifically, on a region of interest (RoI) associated with a person, it performs action classification and density estimation for the action's target object location. This produces a 4-d Gaussian distribution prediction, for each

action type, that models the likely relative position of the target object to the person. The prediction is based purely on the human appearance from a previous human classification model. This human-centric recognition branch, along with a standard object detection branch and a simple pairwise interaction branch, form their multitask learning system. Figure 1 below shows the model architecture proposed by the FAIR paper "Learning to Detect Human Object Interactions"[3].

InteractNet is validated on the HICO-DET and V-COCO data sets.

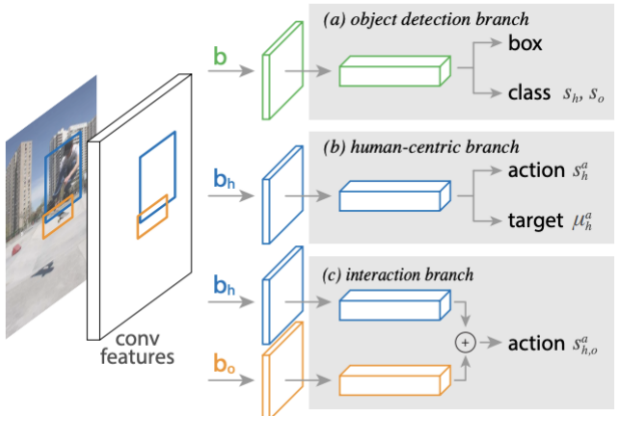

Figure 1. Model Architecture of InteractNet

2.2. Detectron2

Aspects of FAIR's InteractNet[1] is fundamentally built on the existing model Detectron2. Detectron was originally in Caffe2 and was rebuilt using pytorch into Detectron2. Detectron2 is an open source state of the art object detection system developed to help FAIR iterate faster on AI/ML research projects. It is able to draw annotations on images, run benchmarking and evaluations, load model backones, and configure model architectures like the Region of Interest Heads and Feature Pyramid Network. Some examples of object detection tasks it can perform are: object classification with bounding box, semantic and panoptic segmentation, person and keypoint detection and pose estimation.

2.3. HAKE: Human Activity Knowledge Engine

The purpose of HAKE is to build a semantic knowledge engine to help classify human object interactions. HAKE takes a bottoms up approach instead of a top down approach by classifying all parts of the body (head, hand, arms, legs, feet) and establishing links to those parts to the action being performed by a human. For example, labeling a person's left hand on steering wheel, right hand on steering wheel, head inspecting rear view would result in an overall triple label of <human, driving, car>[4]. While this method takes much more labeling labor, by establishing these relationships between all body parts in the image, the model is more robust in identifying actions correctly given different image conditions and variations. The HAKE feature extractor is called Activity2Vec and this feeds into a "part based reasoning network". Figure 2 is an example of the part-state network (PaStaNet) created by HAKE before it is run through a semantic layer of the network to identify the action.

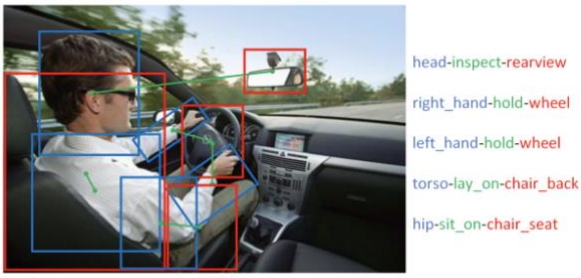

Figure 2. Sample Annotation of Part States in HAKE

2.4. V-COCO

In order to train our action classifier, we leveraged the work of S. Gupta, J. Malik et al in their paper "Visual Semantic Role Labeling"[2] where the Verbs In COCO (V-COCO) data set was introduce. This starts with the object classifier dataset MS-COCO "Common Objects in Context" [8]. Gupta identifies still images from the COCO dataset which involve actions and objects and annotates and labels 10K+ images. We used Gupta's github repo to reconstruct his labeled action data from the COCO set - renamed "V-COCO" to obtain labeled action data which we could then use for our application. Of the labeled verbs, ones of particular interest for our grocery store application were: carry, hold, point, eat, drink, stand, talk on phone.

2.5. HOTR

The End-to-End Human-Objection Interaction Detector with Transforms (HOTR) by Kakaobrain[5] is a novel framework that predicts triplets by implementing a transformer based encoder, decoder, and recomposition layer. Since this project utilized V-COCO data in order to produce HOI triplets, we used this model as our benchmark comparison for our performance in our model. It uses the same action classification verbs which makes comparison simpler when comparing results.

3. Datasets

3.1. COCO & V-COCO

The data we used for this experiment was the V-

COCO dataset built off the 2014 COCO train/val dataset. The V-COCO set consists of 16k+ people instances in 10k+ images. After choosing 25-30 common verbs, V-COCO utilized Amazon Mechanical Turk to generate labels and annotations such as bounding boxes. There is a labeled triple in the form of < the agent (pink box), the instrument (blue box), the object (orange box) >[2].

See Figure 3 for an example annotated photo excerpted from "Visual Semantic…." paper by S. Gupta[2] showing the triplet assigned.

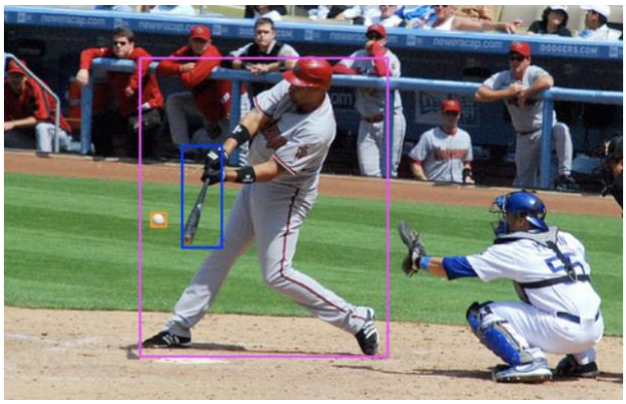

Figure 3. <**agent**: person, **instrument**: bat, **object**: ball>

We decided to use the V-COCO data to implement our model pipeline because Detectron2 is able to register data in the COCO format and produce training and evaluation pipelines that are exportable and repeatable. We felt this would be important as we worked across devices on several candidate model branches for our action detection pipeline.

3.2. MVTech Densely Segmented Supermarket Dataset (MVTech D2S)

The MVTech Supermarket data set[6] is a segmented, labeled, and annotated data set of groceries and everyday items in 60 categories. When using Detectron2, an object detector on some grocery images we scraped from the web, we found that while it could classify objects in the grocery store, it did not necessarily do it correctly. There were only a few food items trained well in the detectron default model so we found the breadth of the food classifications was low. In order to augment this data set for better results, we took the MVTech D2S dataset to further train the Detectron2 model in hopes of improving the accuracy for our grocery store application.

3.3. Human in Supermarket Test SetCOCO

In order to accurately asses whether or not our implementation would be useful in the grocery store setting, we scraped various publications, blogs, and search engines for image data of people in supermarkets doing common actions. Specifically, we targeted images where people were holding or looking at food. We also included other images where targets are just standing and not interacting with food items. We collected 29 of these photos in total and hand labeled them to get metrics for our results.

4. Method

Creating a human object interaction detector requires detecting humans, objects, and understanding the actions in between them. In order to accomplish this, we created a 4-branch inference pipeline to generate a classification for every item in the triplet < human, action, target >. While we took the work from FacebookAI InteractNet as inspiration, we were not able to recreate this pipeline due to the proprietary nature of their code that prohibited them from publishing it. We did however create our own pipeline based on Detectron2 in order to accomplish our human-object-interaction task. The pipeline we implemented is shown in Figure 4. A detailed explanation of tasks for each Branch (A, B, C, D) follows in the respective sections. As mentioned, FacebookAI Detectron2 has many capabilities but we leveraged its

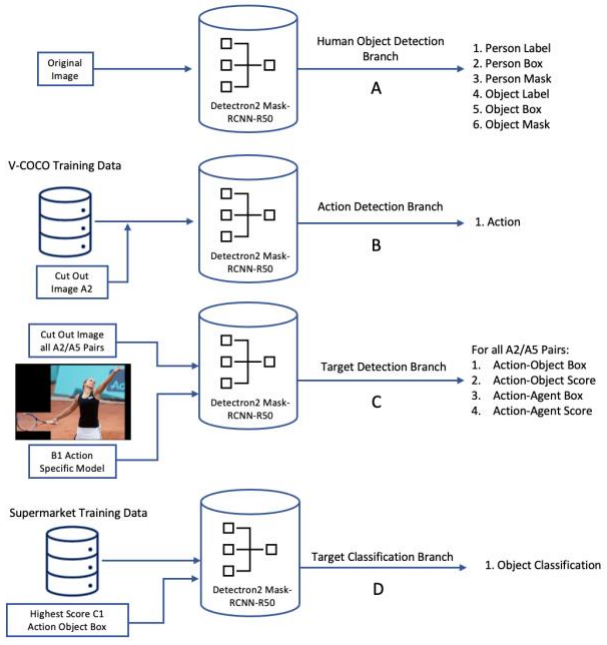

Figure 4. Baseline Architecture of HOI Pipeline

instance segmentation and object detection network backbones in order to train our own customer models. The model we used as our baseline was the Detectron2 Mask-R-CNN model released in 2018. It is a regional based convolutional neural network which generates a bounding box, class label, and an object mask. We chose a model

built with an R-50-C4 backbone because of Resnet-50's good historical performance with image data. We experimented with other models as well, such as the Detectron2 Faster R-CNN and Retinet models, which will be detailed later. However, the Mask-R-CNN served as the baseline for our experiments. We used the default Adam's optimizer and the cross entropy loss solver and did not change these throughout the experiment.

The pipeline consists of 4 branches which generate the HOI triplet. This is accomplished through a human and object detector, action classifier, target detector, and target classifier. For each branch (A, B, C, D) of the pipeline, we obtained model metrics based on a test set which we kept separate from the training set.

We also created a toy data set of our own by scraping Google for "grocery shopping" images. We labeled this "human in supermarket" set. Since this data was manually labeled and too small, we were not able to create meaningful validation metrics on it. However, we do use it to validate the accuracy of our results and provide an example output of each branch of the pipeline.

4.1. Branch A - Human Classification Implementation with Detectron2

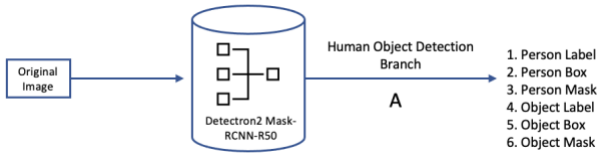

Figure 5. Branch A Diagram

The first branch in our inference pipeline in the Human/Object Classification Branch. In this branch, we detect each human and object in the scene and create a human/object pair for each detected instance in the scene. For example, if this branch finds two objects and two humans, the final output of this branch will be a pairwise list of length four matching each of the objects to each of the humans detected. This pairing is used for inference in branches further downstream.

We utilize the Detectron2 Mask-RCNN pre-trained on a Resnet-50 backbone for our implementation of human classification and basic object detection. Detectron2 baseline is already very high performing for instance segmentation and human recognition, so we did not need to do any more training or transfer learning to get it functioning properly and were able to use the object classification and instance segmentation defaults to accurately find humans in the picture. See Figure 5 for a sample output from the Detectron2 human and object classification. The Mask-RCNN outputs boxes, labels, and masks like the white overlay for the person and the yellow overlay for the skateboard. We use the bounding boxes generated from this model to create human - object pairs that are inputs into our pipeline downstream at Branch C - Action Target Detection.

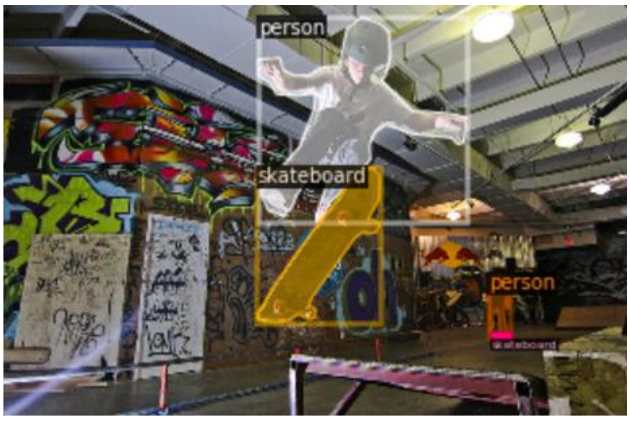

Figure 6. Detectron2 Branch A Output

4.2. Branch B - Action Classification Implementation with V-COCO

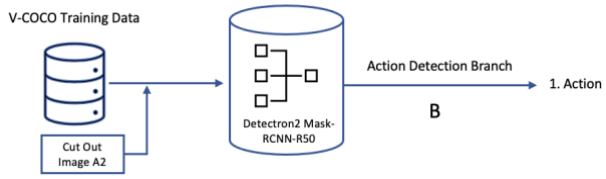

Figure 7. Branch B Diagram

The second branch in the inference pipeline of the Action Classification Branch. This branch takes the person detections from Branch A and classifies it into one of the relevant actions within our dataset. The purpose of this branch is to establish the action that the human is doing using only the visual cues of the person detection. This model does not produce an output that is used downstream and solely produces the action label for the input image.

In order to implement Branch B of our architecture, we trained several model architectures on action classification data. Unfortunately, Detectron2 does not support action classifications by default, so we developed a separate training set from the V-COCO data that correctly labeled person detections as their action label. We put this transformed data set into COCO format so that it could be registered within Detectron2 and used for training and evaluation. As previously mentioned, V-COCO data is verb data labeled in the COCO style. Our starting point for obtaining the V-COCO data discussed in the paper "Visual Semantic Role Labeling"[2] was this github repo. Unfortunately, the data that was labeled in the process of

the mentioned paper was not readily available as a download with the image and annotation pairings, so we had to do some work to create it. First we had to download roughly 19 GB of data: 2014 COCO Train set & 2014 COCO Val Set[7]. While we used the repo as a starting point, it was written in Python 2.7 which we were unable to run in our Google Colab runtime. We used a 2to3 converter to upgrade the scripts from python2.7 to python3, as well as manual hand edits to the code. The annotations were available on the repository but they were not linked to the actual corresponding images, so we had to run a script in order to pair up the images and annotations from the 2014 COCO set that made it into the V-COCO set.

We then used seven classes from this data set that we found to be relevant to grocery shopping, totalling to 1951 images and 4770 instances total. Due to sparse data for some of the classes, for the remainder of the experimental pipeline we focused primarily on using the "hold" action class. "Hold" is both most relevant to grocery shopping observations, and has the most data.

Category	Instances
hold	1838
eat	301
talk on phone	167
carry	236
point	15
drink	63

Table 1. Verb COCO Data

The image input to this Branch B of the pipeline was a slightly modified image from Branch A. Branch A outputs a person bounding box (output item A.2.). Operationally, we then took the image content from inside the person bounding box and inputted only that content into Branch B. This ensures that the action being identified belongs to the person identified from the human object detector. An output of this branch can be seen in Figure 8.

Figure 8. Branch B Output

4.3. Branch C - Target Detection Implementation with Detectron2

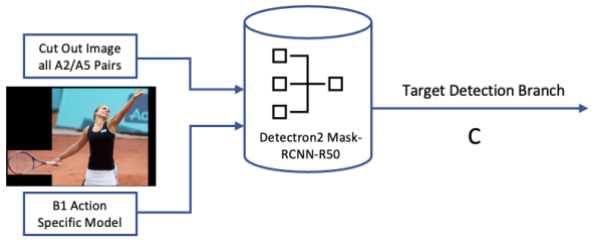

Figure 9. Branch C Diagram

Branch C is the target detection branch. Target detection is the most complex task. The challenge with target detection is that there are many objects in each picture, however not all of them will be the target object the human is interacting with. This requires more work to identify the target correctly from all the objects in the picture. We found there were two main approaches to target detection/classification tasks.

Method 1: Determine the most likely target first, and then classify it with the object classifier.

Method 2: Classify all objects, then determine which object is most likely to be the target.

We took the former approach, however some other state of the art methods take the latter approach as well. However, these state of the art methods usually involve a semantic layer of understanding in a neural network or keypoint analysis or pose analysis. We used a method called agent-object detection. This assumes that for each verb (like "hold") there is a "hold-agent" and a "hold-object".

Figure 10. Agent-Object Cut Out

We have two inputs into Branch C in order to predict the agent/object. The first is a cut out image of every human-object pair in the image (see Figure 10). The cropped image is created from the bounding boxes in Branch A (outputs A.2. and A.5.). The second input is the "action classification" output from Branch B (B.1.). The agent-object scores are based on a specific action, so a separate model must be created for each action. We then take the highest score for "action-object" to determine the correct target. A sample output shows the scores for "hold-agent" and "hold-object" in Figure 11.

Figure 11. Hold Agent-Object Pairs

4.4. Branch D - Target Classification Implementation with Detectron2

The final branch in the inference pipeline is the Target Classifier. The purpose of the target classifier is to correctly identify the agent's action target with a classification that is relevant to the domain in which the inference platform is being deployed. For example, in our grocery store application, the Branch D would be trained to detect food and products of the store. HOwever, in a retail fashion application, this branch would be retrained in order to classify different articles of clothing such as shirts, dresses, or shoes. The purpose of this branch is to make the overall pipeline domain specific and produce relevance results for its industry/application.

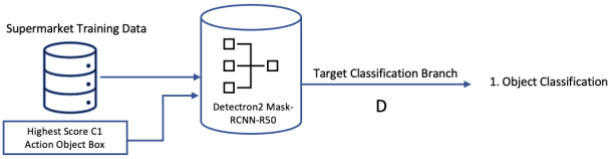

Figure 12. Branch D Diagram

We initially utilized the Detectron2 Mask-RCNN pre-trained on a Resnet-50 backbone for this branch's implementation of object classification for non-human object. We use the instance segmentation to obtain a label, bounding box, and mask of the predicted object. We ran into the difficulty that the object identifier was good in general, however it was not very accurate in classifying specific grocery items. See a sample output below in Figure 13 from the baseline Detectron2. In order to address this, we used additional data from MVTech[6], a dataset labeled in the COCO format, on 60 supermarket categories. We experimented with several models and training methods to improve the results of our target classification Branch D.

Figure 13. Branch D Output

5. Results

The widely used metrics for measuring success in both object detection and human-object-interaction are Average Precision (AP) and Mean Average Precision (mAP).

Average Precision is defined as the average precision for one class where precision is defined as follows:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

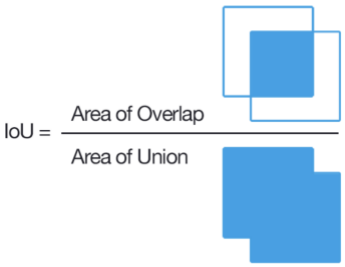

Figure 14. *IoU*

When calculating average precision for object detection with bounding boxes or segmentation, one popular mechanism is to use the IoU (intersection over union) about a certain ratio, such as 0.5 or 0.75. This is represented in the results as AP50 and AP75.

The mean average precision is defined as the average precision over all classes measured:

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{|TP_c|}{|FP_c| + |TP_c|}$$

mAP is also one of the primary metrics of success used for Human-Object-Interaction Detection.

5.1. Branch A - Human Object Detection Branch

For the human-object detection task, used the pre-trained Detectron2 Mask-RCNN R-50 model due to its demonstrated high performance characteristics.

	AP^{bb}_{person}	AP^{mask}_{person}	AP^{kp}
Faster R-CNN	52.5	-	-
Mask R-CNN, mask-only	**53.6**	**45.8**	-
Mask R-CNN, keypoint-only	50.7	-	64.2
Mask R-CNN, keypoint & mask	52.0	45.1	**64.7**

Table 2. Mask-RCNN Results[8]

Table 2 shows the benchmark results of the Mask R-CNN model for multitask learning (box, label, mask) which improves upon the Faster R-CNN when looking specifically at person bounding boxes and person masks. This was trained on the Resnet-50-FPN backbone which we utilized as well. These results are very good and outperform all other models competing in the 2016 COCO challenge[8].

We ran this model on our 29 toy images from the "Human in Supermarket" set and placed the results in "Channel A Outputs" in the github. For all annotations we used a threshold of 0.5 for showing classified results.

5.2. Branch B - Action Classification Branch

For the action classification task, we experimented with several models. We used COCO 2014 data for our V-COCO training and validation sets to measure the performance of the models. The first experiment was on Detectron2's Mask-R-CNN R-50 model trained on the V-COCO labeled data with a learning rate of 0.002 and 1000 iterations. Next we trained on the Faster-R-CNN R-50 with the same data set, a learning rate of 0.002 and iterations of 1000. Next we trained on the Retinanent R-101 model at a learning rate of 0.002 and 1000 iterations. A summary of the AP results can be seen below in Table 3.

Model	`mask_rcnn_R_50`	`faster_rcnn_R_50`	`retinanent_R_101`
AP	11.216	10.818	14.973
AP50	18.607	16.965	22.975
AP75	12.625	12.392	16.646
APL	12.547	11.430	14.828
mAP	12.010	15.232	18.309
"Hold" AP	34.787	39.873	37.947

Table 3. Branch B Model Metrics

In the table above, the two metrics which interested us most were mAP (mean average precision) and "Hold" average precision. Mean Average Precision measures the average precision of all 7 classes. "Hold" Average precision is just for the "hold" class, which is the one we were most interested in. We chose the Retinanent-R-101 model with a learning rate of 0.002 and 1000 iterations as our best performing model. Even though it did not have the best "Hold" average precision, it had the best mean precision of 18.309, so we thought it would generalize best to new image sets. The performance of the "Hold" class alone was still pretty good at 37.947 and was the second best in this category. We ran this model on our 29 toy images from the "Human in Supermarket" set and placed the results in "Channel B Outputs" on the github. For all annotations we used a threshold of 0.5 for showing

classified results.

5.3. Branch C - Target Detection Branch

For the target detection task, we experimented with several models. We used COCO 2014 data for our cutout V-COCO training and validation sets to measure the performance of the models. The first experiment was on Detectron2's Mask-R-CNN R-50 model trained on the V-COCO labeled data with a learning rate of 0.002 and 1000 iterations. Next we trained on the Faster-R-CNN R-50 with the same data set, a learning rate of 0.002 and iterations of 1000. Next we trained on the Retinanent R-101 model at a learning rate of 0.002 and 1000 iterations. A summary of the AP results can be seen below in Table 4.

Model	**mask_rcnn_R_50**	**faster_rcnn_R_50**	**retinanent_R_101**
AP	12.593	12.776	7.337
AP50	22.356	20.392	13.567
AP75	13.757	14.800	7.176
APL	16.826	14.958	6.352
mAP	13.496	14.092	9.129
"Hold_obj" AP	26.770	38.219	8.012

Table 4. Branch C Model Metrics

Again, the two metrics which interested us most were mAP (mean average precision) and "Hold_obj" average precision. Mean Average Precision measures the average precision of all 7 classes. "Hold_obj" Average precision is just for the "hold_obj" class, which is the one we were most interested in. We chose the Faster_rcnn_R_50 model with a learning rate of 0.002 and 1000 iterations as our best performing model. We ran this model on our 29 toy images from the "Human in Supermarket" set and placed the results in "Channel D Outputs" on the github. For all annotations we used a threshold of 0.75 for showing classified results.

5.4. Branch D - Target Classification Branch

For Branch D, our target classification started with a baseline of the Detectron2 Mask-R-CNN R-50. While this model is highly accurate for human detection and some general object detection, the number of food classes is small. The only foods represented are carrot, donut, cake, orange, banana, apple, broccoli, and pizza. This is only 8 classes, so we found it necessary to further train the model. The MVTech Densley Segmented Supermarket data set contains COCO-style labeled images from 60 categories. We trained a model with this data and validated on this data set too. We also measured the accuracy of the baseline Detectron2 Mask-R-CNN performance on this MVTech validate set as well.

We trained a Mask-R-CNN R-50 model on the MVTech data with a learning rate of 0.002 and 400 iterations. We trained a Retinanent R-50 model on the MVTech data with a learning rate of 0.002 and 400 iterations.

Model	**Detectron2 faster-rcnn_R_50**	**MVTech mask_rcnn_R_50**	**MVTech retinanent_R_50**
AP	7.071	5.271	7.072
AP50	10.595	10.472	8.444
AP75	7.809	4.237	7.880
APL	7.080	5.277	7.114
mAP	0.623	0.554	4.175

Table 5. Branch D Model Metrics

After seeing the results of the Detectron2 default Faster-R-CNN being better in some areas than the MVTech trained models. However the mAP was still extremely low. We suspected that the Faster-R-CNN was a more performant model. So we re-ran the MVTech trained data on the Fast-R-CNN model and as expected, obtained better results. We upped the iterations from 400 to 1000 on a second run and obtained results we were happy with for

these purposes given the breadth of classes we were working with.

Model	MVTech faster-rcnn_R_50 - 400 iter	MVTech faster-rcnn_R_50 - 1000 iter
AP	9.632	23.377
AP50	18.417	43.035
AP75	8.629	23.494
APL	9.637	23.442
mAP	2.353	9.414

Table 6. Branch D Faster-R-CNN Model Metrics

5.5. HOTR Benchmark Results

To compare our performance to past work, we compared our human-object-interaction detection pipeline with results with other state of the art HOI detectors to get a relative gauge of performance. The detector we chose to compare to is the End-to-End Human Object Interaction Detection with Transformers (HOTR) pipeline as it trained on the V-COCO dataset and published its AP Hold results. HOTR was able to achieve an AP of 48.98 for the "Hold" action triplets. While this is higher than our AP = 38.22 for Hold, we believe these results are encouraging and speak to the potential of our method in identifying HOI triples.

6. Conclusion

Determining human-action-target triples is a challenging task that requires hyperparameter optimization and multiple model architectures in an inference pipeline in order to work effectively. Based on the model results obtained from our experiments, we determined that multiple data sources training multiple model architects within an inference pipeline was the best way to accomplish the task of generating human interaction data for retail stores.

In summary, our method of determining human actions and action-targets without semantic data proved to be a useful implementation that warrants more development. Although our full pipeline's performance (AP = 38.22) fell short of HOTRs baseline performance (AP = 48.98), we believe that interpreting human actions entirely from their visual cues as demonstrated is a promising. With more training data, GPUs, and training time even better results could be achieved. These methods can certainly progress the field of data collection in in-person retail stores to and observe their customers spending habits effectively.

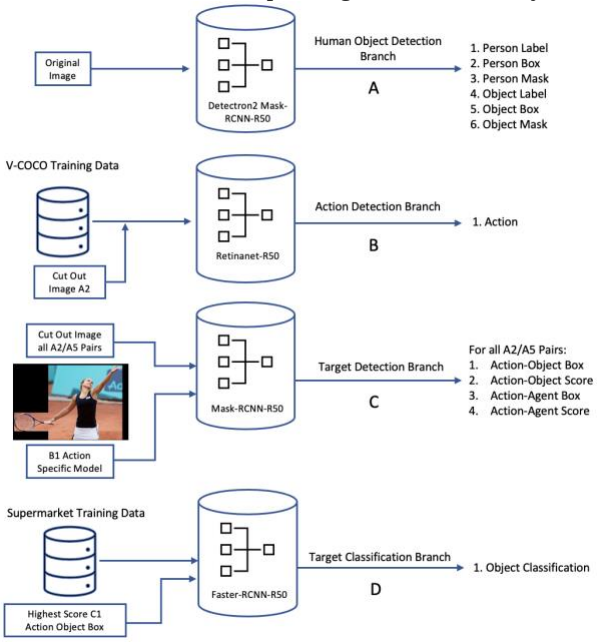

Figure 15. Inference Pipeline

7. References

[1] Gkioxari, Georgia, et al. "Detecting and recognizing human-object interactions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

[2] Gupta, Saurabh, and Jitendra Malik. "Visual semantic role labeling." *arXiv preprint arXiv:1505.04474* (2015).

[3] Chao, Yu-Wei, et al. "Learning to detect human-object interactions." *2018 ieee winter conference on applications of computer vision (wacv)*. IEEE, 2018.

[4] Li, Yong-Lu, et al. "HAKE: A Knowledge Engine Foundation for Human Activity Understanding." *arXiv preprint arXiv:2202.06851* (2022).

[5] Bumsoo Kim, et al. "HOTR: End-to-End Human-Object Interaction Detection with Transformers." *CVPR*. IEEE, 2021.

[6] Patrick Follmann, Tobias Böttger, Philipp Härtinger, Rebecca König, Markus Ulrich: MVTec D2S: Densely Segmented Supermarket Dataset; in: *European Conference on Computer Vision (ECCV), 569-585, 2018.*

[7] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." *European conference on computer vision*. Springer, Cham, 2014.

[8] He, Kaiming, et al. "Mask r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2017.