Analysis of Machine Learning Models for Trash Detection

1st Jerome Newhouse
newho081@umn.edu

2nd Ryan Roche
roche189@umn.edu

3rd Isaac Berlin
berli113@umn.edu

4th Robert Wang
wan00379@umn.edu

***Abstract*—This paper explores the application of computer vision and machine learning techniques for automated waste detection and segregation, addressing a critical environmental and public health issue in developing countries. First, we aim to improve waste sorting efficiency by training models to detect different types of trash and recyclables, which can potentially automate the sorting process, reducing human error and increasing the overall recycling rate. Second, we develop and evaluate four different models, GrondingDino, Fine-Tuned DETR, YOLOv8, and ResNet, using the TACO dataset from Kaggle, which contains images of waste in real-world settings. Finally, we discuss some of the limitations and challenges we faced along the way.**

I. Introduction and Motivation

A. Introduction

The improper sorting of waste is a significant environmental challenge in the United States. With only 2 out of 5 Americans properly sorting their trash and a mere 30% of recyclable materials being correctly processed, there is a clear need for innovative solutions to improve waste management practices. Our project aims to address this issue by developing and evaluating machine learning models for automated garbage and recycling detection.

B. Motivation

The motivation for this project is twofold; first, we aim to improve waste sorting efficiency by training models to detect different types of trash and recyclables, which can potentially automate the sorting process, reducing human error and increasing the overall recycling rate. Since there is existing research on waste detection our goal for the project is to train, evaluate, and compare multiple state-of-the-art models to determine the most effective approach for this specific task.

C. Report summary

In our report, we first highlight the previous work that has been done in trash detection and classification. Next, we talk about our approach to our problem, the dataset we used, the evaluation metrics, as well as the models we plan to use: GroundingDINO, DETR, YOLOv8, and ResNet. For each model, we then have comprehensive recounts of the experiments and results. In addition, we also highlight our bonus feature which is an app that utilizes our YOLO model to be able to detect trash on your phone. Finally, at the end of our paper, we have our findings and compare the best results from all the models to determine which model did the best on this task as well as talk about some of the limitations and challenges we faced along the way.

II. Related Works

We have reviewed several papers related to waste classification using computer vision and machine learning techniques. Here, we summarize three prominent works that will inform our approach.

A. Computer vision for solid waste sorting: A critical review of academic research

One of the academic papers that we reviewed was "Computer vision for solid waste sorting: A critical review of academic research" by Weisheng Lu and Junjie Chen[1]. This paper is a comprehensive collection of all research papers on sorting solid waste using different machine-learning techniques. The paper first talks about the 3 main use cases that trained models that could be used to help properly sort trash. These 3 use cases are residential and municipal services (RM), industrial, commercial, and institutional sources (ICI), and construction and demolition (C&D) activities. They then look into the history of this problem and the amount of research papers from 1997 to 2021 has increased from about 1 paper a year to around 8 papers a year.

The main part of the paper is breaking down the different machine learning models that have been used by researchers to try and solve this problem. They broke these machine learning models into two categories; traditional and end-to-end deep learning. The traditional models consisted of models like support vector machines, nearest neighbors, and decision trees. While the end-to-end deep learning models consisted of models like AlexNet, ResNet, and Faster R-CNN. For each model, they give a quick description of the model and provide the results researchers had when using those models.

In addition to breaking down the types of models researchers have used they also talk about the datasets that they have used to train their models. The majority of the datasets that researchers used were private datasets that the researchers created themselves. However, they did provide 10 datasets that were used that are publicly available and gave some details about the datasets like what type of waste was present and what kind of task it would be used for; detection of multiple pieces of trash, and recognition of trash in the image.

The most interesting part of this paper was at the end when they talked about what needs to be done in the future to help

improve future models as well as improve waste sorting. The first suggestion was to create public industry-specific datasets. As mentioned before the majority of the research was done using private datasets, but creating public datasets for a specific industry would allow better performance in specific use cases as well as be able to compare the performance of different models on the same dataset. They also pointed out the uneven distribution of research in use cases. There have only been 11 research papers for sorting construction and demolition waste of the past two decades. They said that there needs to be more effort and data to target this use case as 40% of the waste in the current waste stream comes from construction and demolition.

B. Development of Computer Vision Algorithms for Multi-Class Waste Segregation

[2] This paper explores the application of computer vision algorithms for automated waste detection and segregation, addressing a critical environmental and public health issue in developing countries. The accumulation of unsorted waste in landfills poses significant challenges, including space constraints and health hazards for manual waste sorters. The research proposes an innovative solution: smart waste bins equipped with computer vision capabilities to segregate waste at the source.

The advantage of source segregation over large-scale separation at landfills lies in its potential for higher accuracy and efficiency. By classifying waste items individually at the point of disposal, the system can minimize errors caused by waste deformation or occlusion that often occur in landfill environments. While the initial cost of implementing multiple smart bins may be higher, the long-term benefits in terms of accuracy and reduced health risks are substantial.

The study compares three prominent computer vision neural networks: Convolutional Neural Network (CNN), You Only Look Once (YOLO), and Faster Region-Based CNN (R-CNN). Each algorithm presents a unique set of strengths and limitations:

1) CNN: While the least accurate, it offers faster processing and requires less training data, making it suitable for scenarios with limited computational resources or smaller datasets.
2) YOLO: Strikes a balance between speed and accuracy, performing well in real-time applications. It requires a larger training dataset but processes images more quickly than R-CNN.
3) Faster R-CNN: Achieves the highest accuracy but demands the most extensive training data and computational power. It's ideal for applications where precision is paramount and processing time is less critical.

The results indicate that YOLO and Faster R-CNN outperform CNN in terms of accuracy, with Faster R-CNN showing the best overall performance. However, the choice of algorithm depends on the specific requirements of the implementation, considering factors such as available computational resources, dataset size, and the balance between speed and accuracy needed for the application.

Building upon the findings of this study, future research could explore several key areas to enhance the effectiveness and practicality of computer vision-based waste segregation systems. A promising direction is the integration of multi-modal sensing techniques, combining computer vision with other sensor types to improve classification accuracy for challenging materials. Additionally, investigating the application of edge computing could optimize real-time performance, enabling faster and more efficient on-site processing. To address the dynamic nature of waste composition, developing adaptive learning algorithms would allow the system to continuously update and improve its classification capabilities. Crucially, large-scale deployment studies in diverse urban environments would provide invaluable insights into the system's real-world performance and scalability. These advancements, coupled with an expanded focus on assessing recyclability, could significantly contribute to the broader goal of creating more sustainable and efficient waste management practices worldwide.

C. An Efficient Multi-Label Classification-Based Municipal Waste Image Identification

"An Efficient Multi-Label Classification-Based Municipal Waste Image Identification"[3] by Wu et al. extends the idea of trash and recycling detection to a street view level. The motivation for this paper centered around the growing power of computer vision classification techniques and the general problem that piling up garbage causes. It explains that as populations grow they produce more waste, and we need to work towards more efficient waste management to promote the cleanliness and health of urban environments.

This paper focuses on a model and dataset. The model, titled Query2Label, is a variation of a vision transformer (ViT) which utilizes self-attention and cross-attention with an asymmetric loss function. The dataset, titled Garbage in Garbage Out (GIGO), is a dataset containing 25,000 images of streets in Amsterdam with varying levels of garbage on them. These images were labeled as Not Garbage, Garbage, Garbage Bag, Cardboard, Bulky Waste, and Litter. The images also had varying levels of garbage in each image from zero for the Not Garbage class up to 4 combined features.

The implementation of the ViT in this paper was fairly standard and straightforward. Each image was broken into 16x16 pixel patches, flattened, and then ran through multiple attention masks and forward layers. One interesting feature of this paper implemented was the asymmetric loss function. This loss function is similar to cross-entropy but adds a focusing parameter for positive and negative predictions which allows the model to be more sensitive to uncommon categories.

The outcome of this paper is that using the methods stated above, the researchers were able to perform around 3 to 5 percent better than previous methods (ResNet, MobileNet) on the classification of their dataset. They were also able to complete this task with less training FLOPS required compared to other methods. This paper provides an interesting insight into the practical applications of how garbage classification can be

implemented in the real world and some of the constraints it faces. It also shows that transformer-based vision techniques can perform at the same level as standard neural network-based models.

III. Our Approach

A. Models

For our project we focused on training and evaluating four different models using a common dataset to compare their performance in waste detection tasks. The models we trained and used are as follows:

1) GroundingDINO: A transformer-based model designed for analyzing natural language queries in visual scenes, enabling object detection and localization based on textual descriptions.
2) Detection Transformer (DETR): A deep learning model that combines Convolutional Neural Networks and uses Transformers.
3) YOLOv8: A fast and efficient model that predicts classes and bounding boxes for multiple objects in real-time from a single pass over the image.
4) ResNet: A residual network (CNN) designed to enable deep networks by using residuals to mitigate the vanishing gradient problem during training.

B. Data

We used a dataset called Trash Annotations In Context (TACO). This dataset contains 11,910 images of size 416x416. The data was split into test, train, and validation splits of 100, 8401, and 3409 images respectively. The dataset contains images of trash in a real-world setting. The dataset contains 18 classes of garbage ranging from cans and bottles to broken glass and cigarettes.

C. Evalution Metrics

To evaluate the model's performances against each other we use precision, recall, mAP50, and mAP50-95 as our evaluation metrics. We chose these metrics as they are standard evaluation metrics used when evaluating object detection models.

Precision measures the proportion of correctly predicted positive detections out of all predicted positive detections, defined as $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives+False Positives}}$. *Recall* quantifies the ability of the model to detect all relevant objects, calculated as $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$. The *mean Average Precision at IoU threshold 0.5* (*mAP@50*) is the average precision computed across all classes, using a threshold of 0.5 for the Intersection over Union (IoU) between predicted and ground-truth bounding boxes. Meanwhile, *mAP@50-95* is a more comprehensive metric that averages the precision across IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05, providing a more detailed assessment of model performance.

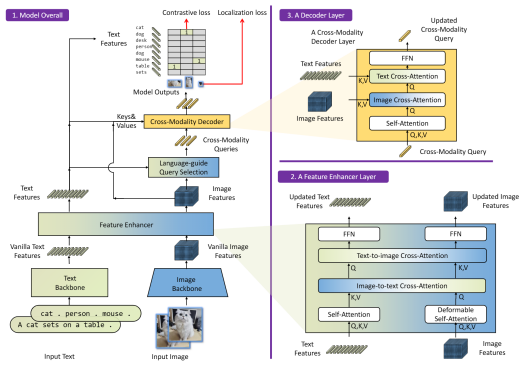

Fig. 1. GroundingDINO model architecture

IV. Experiments and Results

A. GroundingDINO

GroundingDINO is a popular visual-language model for object detection and segmentation [4]. Unlike closed-set object detectors, which can only detect objects in pre-defined categories explicitly given in the training data, GroundingDINO is able to detect objects in an open set of categories. The model is able to effectively "free" itself from closed-set class restrictions by utilizing a BERT text encoder [5] to help generalize from the closed-set training data to novel descriptors. When a text and image prompt is given to the model, both the image and text backbones individually extract information. Cross-attention then combines the embeddings to be used for language-guided query selection, wherein the model negotiates sections of the input image that it believes to be relevant to the text prompt, and vice-versa. The "raw" embeddings are then decoded into bounding boxes, and text tokens from the prompt are associated with each bounding box.

IDEA-Research, the developers of GroundingDINO, haven't yet released their training code for the model, so it wasn't possible to train weights on any specific dataset. For evaluation, we tested the model with their provided weights, trying different combinations of prompts and hyperparameters. The main two hyperparameters are the 'box_threshold' and 'text_threshold' values, which enforce minimum values on how confidently the model believes a proposed box location is correct, and how confidently it believes a detected region is to the text prompt, respectively.

To evaluate the GroundingDINO model, a grid search approach was taken, where combinations of different hyperparameter values were tested, with inference performed using multiple text prompts of varying specificity on each hyperparameter combinations. Combinations of the values 0.25, 0.35, 0.5, and 0.75 were used for the 'box_threshold' and 'text_threshold' values. For the prompts, four levels of specificity were tested. Ordered from least specific to most specific, they were as follows:

Prompt Name	Prompt Content
very_generic	”objects”
supercategories	All supercategory names from the TACO dataset, period-separated. i.e. ”Bottle.Bottle cap.Broken glass.Can...”
categories	All category names from the TACO dataset, period-separated. i.e. ”Other plastic bottle.Clear plastic bottle.Glass bottle...”

Inference ran at roughly 2 FPS on an NVIDIA RTX A5500 workstation GPU, indicating that the GroundingDINO model is not yet suitable for real-time applications, especially not on mobile devices. It is possible that inference could be performed quickly enough for real-time use by deferring the calculations to some sort of cloud compute platform, but that is beyond the scope of the resources available to us.

After running and saving the results of inference on each image in the TACO dataset with all aforementioned combinations of hyperparameter values and prompts, the performance of the model was evaluated using the metrics described in the Evaluation Metrics section of the paper.

While preliminary testing showed great promise for GroundingDINO's ability to idenity objects in otherwise busy images, it struggled heavily with the images in the TACO dataset. Like the other models, it struggled mostly with the extremely small objects, like cigarettes and bottle caps. However, a problem that appears to be a side effect of the language processing component of the model is its confusion between similarly-appearing objects, such as mistaking a glass jar for a plastic bottle.

In fact, across *all* of the predictions with the best-performing hyperparameters, not a single predicted label matched any of the ground-truth labels for that image! For some of the images, it failed to produce any bounding boxes entirely. This is mostly due to the language confusion described earlier- the model often mistook objects of similar appearances. It wasn't even possible to generate a confusion matrix, as there weren't any predicted bounding boxes that met the IoU threshold to count as a false positive for the ground-truth labeling! Nevertheless, this extremely poor performance gives us more than enough information to all but eliminate GroundingDINO as an option for trash classification, at least in its current state without the ability to perform custom training.

The poor performance of the model with the TACO dataset's classes, combined with the long inference time make it an unlikely candidate for any sort of real-time application. Better performance is likely possible with custom training once the code is made available by IDEA-Research.

B. YOLOv8

The ”You Only Look Once” or YOLO model is a popular deep learning architecture for object detection [6]. It is designed to perform with high detection accuracy while being able to achieve real-time performance. YOLO can predict bounding boxes and class probabilities for an image in one forward pass through the model. YOLO divides the image into a grid and, for each grid cell, predicts bounding boxes and their associated confidence scores. YOLO uses a convolution neural network (CNN) to extract features. The architecture of the YOLO model can be seen in Figure 2 In addition, YOLO uses post-processing techniques like Non-Max Suppression to refine predictions. The design of YOLO allows it to be used in all spectrum of contexts.

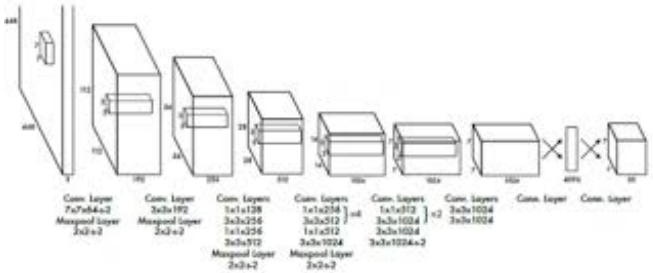

Fig. 2. Architecture of YOLO model.

For testing and training the YOLO model, we used the YOLOv8x model as the base model for all experiments. We first started by training the YOLOv8x model on the TACO dataset to get some initial results.

We used YOLO standard training parameters like a batch size of 16 and the number of workers set to 8. We had to change some parameters like the image size expected to speed up training and inference if we didn't, the model would have to scale the image up to 640 x 640 and then back down to 416 x 416 for every image. For this initial training, we started with 500 epochs.

When training with these parameters and 500 epochs on a 2060 Super it took around 2 days to complete the initial training. Once the training was complete we got a precision of 0.713, recall of 0.427, mAP 50 of 0.483, mAP 50-95 of 0.386. In addition to these metrics, we also tested how fast the inference was and was averaging around 200 FPS or 5 ms. These initial results didn't seem the best as we were hoping for better precision and recall after training. However, after looking at the confusion matrix shown in Figure 3 that is generated from the training it did appear that the model seemed to be doing a good job at detecting and labeling bottles, cans, and plastic bags. However, the model was struggling to identify bottle caps, broken glass, cigarettes, and pop tabs. These findings made a lot of sense to us based on the dataset we used. The images in our dataset were 416x416 which is not the largest and the trash in the images is in a real-world setting so there are no closeups of the objects and ofter farther away from the camera. We believe that this caused some issues with the detection of smaller objects like bottle caps and cigarettes. On the other side objects that were bigger like bottles, cans, and plastic bags were larger and in turn, would show up better on the images seemed to be easier to find due to their size.

When looking at the loss curve of the training it also

Evaluation Metric	500 epoch TACO dataset	1000 epoch TACO dataset	500 epoch modified TACO dataset	200 epoch new dataset
Precision	0.713	**0.777**	0.768	0.435
Recall	0.427	0.398	0.434	**0.442**
mAP50	0.483	0.491	**0.517**	0.433
mAP50-95	0.386	0.403	**0.424**	0.327

TABLE I
EVALUATION METRIC SCORES FOR YOLOV8 MODELS.

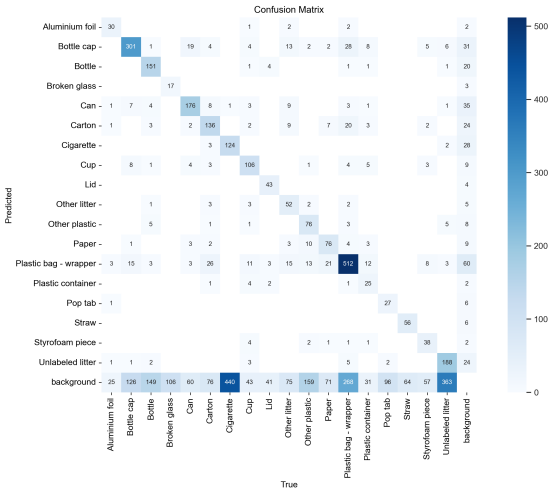

Fig. 3. YOLO confusion matrix from 500 epoch training.

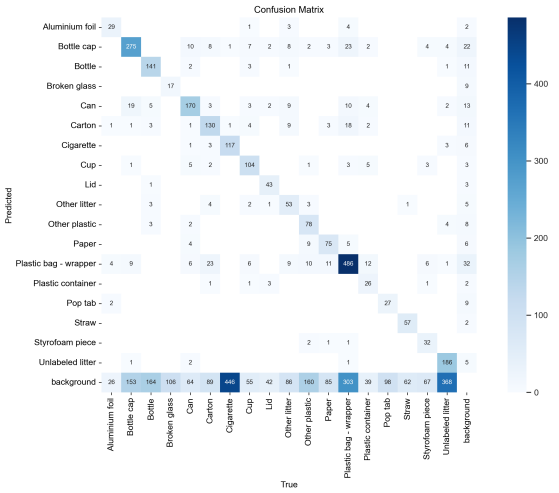

Fig. 4. YOLO confusion matrix from 1000 epoch training.

seemed that the model was still improving slowly so we decided to try training again using 1000 epochs this time. This did result in slightly better results shown in Table I. When training the model for 1000 epochs it took about 4 days but gave a precision of 0.777, recall of 0.398, mAP50 of 0.491, and mAP50-95 of 0.403. These results were somewhat promising as they were slightly better than the 500 epoch training sessions. However, the model still had issues detecting the smaller objects similar to the initial results, and was better at the larger objects.

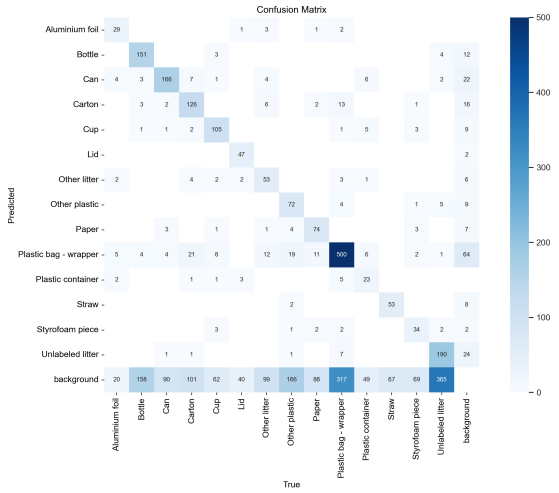

Fig. 5. YOLO confusion matrix from 500 epoch training with small objects removed.

After these initial findings of the model being better at detecting the larger objects in the images compared to the smaller objects, we wanted to see how the model would perform if we removed the annotations for the smaller objects in the dataset. So we removed all the annotations for bottle caps, broken glass, cigarettes, and pop tabs from the annotations. we retrained the model on this "new" dataset without the four small objects that seemed to be the hardest to detect. Similarly to the default model we used the default parameters except for changing the image size to increase inference and training speed. For this training we also had the model train for 500 epochs. Similar to the initial training this took about 2 days to complete the entire training. We were happy to see that removing these smaller objects from the annotations resulted in all the metrics than the original 500 epoch training with the entire dataset. When removing the small objects we got a precision of 0.768, recall of 0.434, mAP50 of 0.517, and mAP50-95 of 0.424 shown in Table I. With half the amount of epochs, I was able to get a similar performance to the 1000 epoch model which I thought was very promising. In addition to the metrics performing similarly to the 1000 epoch training, the model was able to detect the larger objects at the same rate based on the confusion matrix shown in Figure 5.

Even with this improvement, we were still curious to see if better results were achievable. With the TACO data set having images taken in the 'real' world like the beach or the street, we wanted to see what the YOLO model could do if the trash images were closer to the camera and in a more controlled

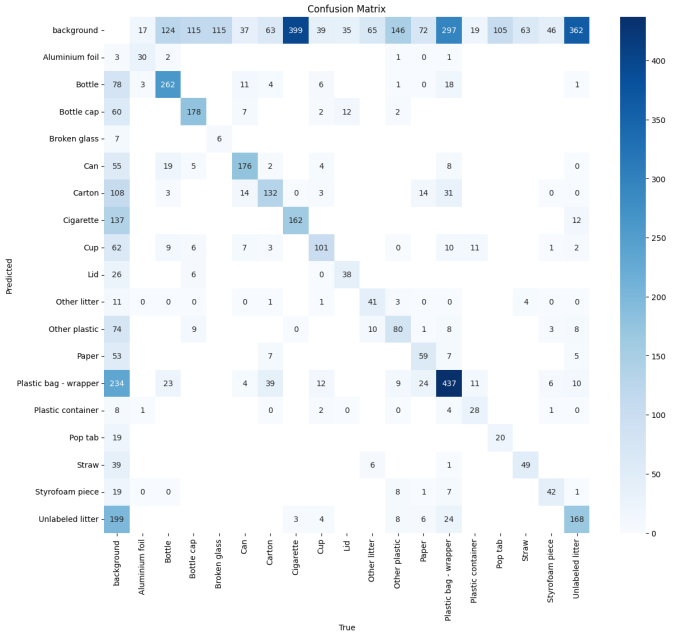

Fig. 6. ResNet confusion matrix with 24 epoch, 4 batch size

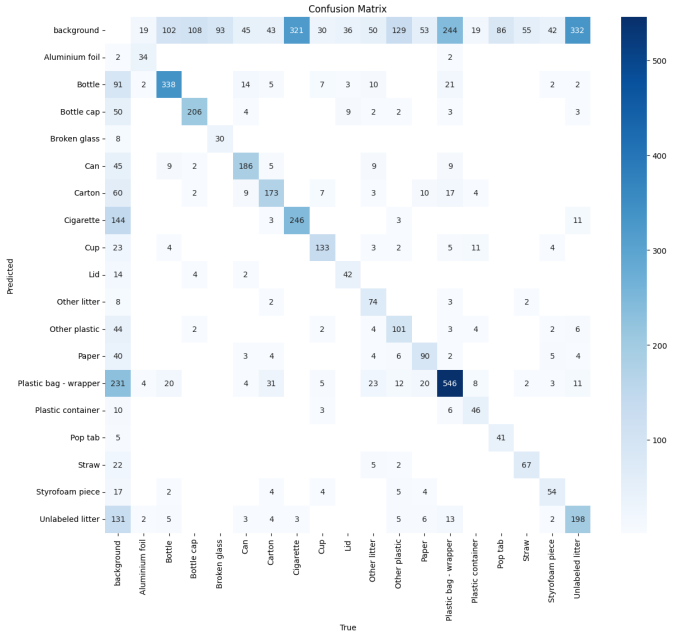

Fig. 7. ResNet confusion matrix with custom sampling and loss

environment. we found a different dataset that had photos of garbage that was closer to the camera. However, this dataset had 43 classes of objects compared to the 18 in our dataset. For this, we used the default YOLO settings and only used 200 epochs as This was just a test to see if having the objects closer would yield better results. Unfortunately, this training ended up yielding the worst results as shown in table I. We were surprised by these results as we thought with the objects being larger in the image would allow the YOLO model to detect the objects more clearly.

In conclusion, the YOLO model does a decent job at detecting different types of garbage with the best results coming from the 1000 epoch model with a reported precision of 0.777, recall of 0.398, mAP50 of 0.491, and mAP50-95 of 0.403 as shown in table I. Throughout all our testing and training with the YOLO model there was a common theme that occurred where the models were good at detecting the larger objects compared to the smaller objects. We think that this is mainly due to the dataset itself. With the images being lower resolution it makes it harder for smaller objects to be detected. We were surprised by the results of the other dataset that we used for our last test. We thought that it would yield better results due to the objects being closer to the camera, but it produced worse results compared to the TACO dataset.

C. ResNet

The next model that we analyzed is a Faster RCNN model with a ResNet-50 backbone. This model is an advanced object detection architecture that combines two key components: a deep convolutional neural network for feature extraction and a region proposal network for identifying potential object locations. The ResNet-50 backbone functions as the feature

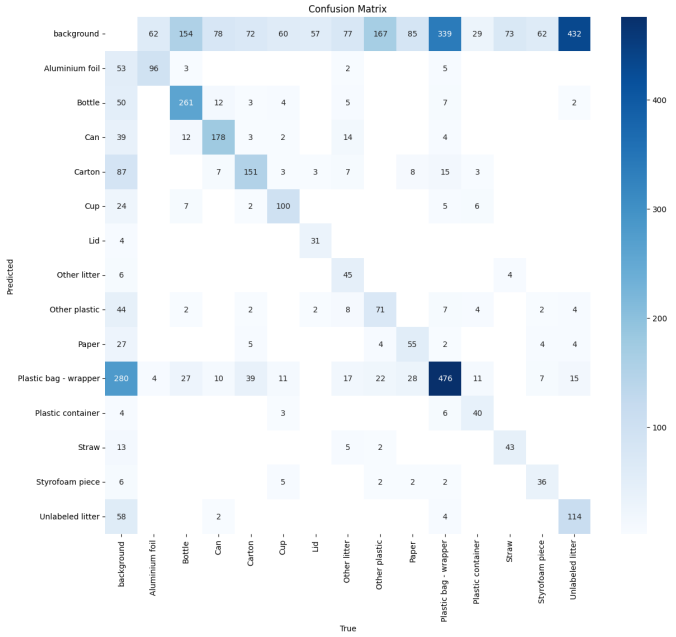

Fig. 8. ResNet with small objects removed

extractor, and the faster R-CNN framework uses these features to generate region proposals, which are then refined and classified, allowing the model to precisely localize them within an image. This approach provides a powerful and efficient method for identifying and bounding multiple objects in complex visual scenes like the one we have for our dataset.

When deciding which ResNet variant to use and whether to train from scratch, we had to consider balancing model complexity, performance, and computational cost. Although

a ResNet-18 backbone would have been faster to train, our dataset includes many tricky classes, some of which are occluded, deformed, or very small, and thus a more capable feature extractor is desirable. On the other hand, moving to ResNet-101 would likely introduce unnecessary complexity and slow down training significantly on our limited hardware. We chose to fine-tune a ResNet-50 pretrained on ImageNet, a large-scale dataset of millions of labeled images across over 1,000 classes [7]. This fine-tuning strategy not only sped up our training process but also improved results compared to training from scratch, because the pretrained weights already capture general visual features useful for detecting waste objects.

Before training, we converted the dataset's YOLO-format annotations to Pascal VOC format, which Faster R-CNN expects. YOLO labels consist of a class ID followed by normalized coordinates and dimensions of the bounding box. In contrast, Pascal VOC uses two coordinate pairs representing the top-left and bottom-right corners of the bounding box. This conversion was straightforward and ensured our model's predictions would align with the required format. We also had to take into consideration the proper indexing of the class IDs, since ResNet expects a "background" class indexed at 0, we had to shift the original indices by 1 in order to account for this.

For this model, we experimented with various hyperparameters. An initial attempt using a high batch size (16–24), about 50–70 epochs, and a 0.01 learning rate, which is similar to the original training rate for ResNet-50, yielded a precision of 0.41, recall of 0.38, and mAP of 0.29. Lowering the batch size (2–4), reducing epochs (20–30), and using a more conservative learning rate (0.001–0.005) produced similar results, with a precision of 0.42, recall of 0.39, and mAP of 0.26 6.

To improve upon these suboptimal outcomes, we tested data augmentation via the Albumentations library, applying rotations, flips, and scaling. While these augmentations slightly improved initial precision, the performance plateaued at similar values to before, with a somewhat higher training loss. Since our dataset already contained a variety of orientations, the additional augmentations did not provide a substantial benefit. More aggressive transformations caused underfitting, indicating the model struggled with overly distorted data.

Examining the confusion matrix revealed that the model performed poorly on small objects, such as tiny pieces of unlabeled litter, broken glass fragments, or cigarette butts, which we defined as having a bounding box area of 500 pixels or less. These small items accounted for over half of the model's missed detections. In contrast, the model excelled at detecting medium-to-large items, even with partial occlusions or overlaps.

Notably, when retraining the model with the same parameters and after removing small object classes like bottle cap, cigarettes, broken glass, and pop tabs the precision increased to around 0.54, demonstrating that these tiny items are indeed a major challenge 8.

To address this issue in another manner, we implemented a custom sampling function to ensure that each training batch included a certain proportion of images containing at least 3 small waste objects. Additionally, we introduced a modified loss function that assigns a higher weight (1.5–2.0 times) to mis-classifications and localization errors involving small objects. These modifications helped the model focus more on these harder-to-detect items. As a result, we improved the model's precision to 0.503 and achieved an mAP of 0.352 7, a notable improvement over our previous results, but still not the ideal outcome. This helps us conclude that the dataset image dimensions are simply not enough to train effectively on the tiny pieces of waste.

D. DETR

The Detection Transformer, or DETR [8], is a transformer based model for object recognition proposed by researchers at Facebook AI in 2020. This model works by using a CNN encode images, flattening this encoding, adding positional embedding, and passing all this into a standard encoder decoder style transformer. The final predicted bounding box and label are then determined by feed forward network. Because of the real time aspect of our bonus feature we originally attempted to work with a Real-Time Detection Transformer or RT-DETR. This model from Zhao et al [9] replaces the CNN backbone with an Efficient Hybrid Encoder of their own design.

We originally investigated training our own DETR and RT-DETR from the ground up using PyTorch. However, given the size of our training data (4.2K) versus the size of the COCO dataset that these models are trained on in their papers (118K) and the large amount of data generally needed to train a transformer model, we decided to fine-tune pre-trained versions of these models.

We chose a DETR and RT-DETR which were both originally trained only on the Common Objects in Context, or COCO, dataset as previously stated above. For the DETR models with a backbone of ResNet-50 and ResNet-101 for the encoding step, and for the RT-DETR model we used their own backbone for the encoding step.

We trained each model for 30 epochs and inspected results, which is where we noticed that none of the above models were able to make any high confidence predictions. We originally assumed errors within our code or data preproccessing, but after a deep dive into our code and exploring the finer details of training transformers we determined that we had not trained for enough time or epochs.

During the debugging of our previously trained models, we found another paper that introduced a faster converging DETR model. This paper, from Microsoft Research Asia [10], introduced the Conditional DETR. This iteration of a DETR showcased a novel cross-attention mechanism which ideally allows of the DETR model to converge in substantially less epochs. The paper displayed that, with this new conditional cross-attention mechanism, their DETR was able to converge within 50 to 100 epochs as opposed to a traditional DETR which converged within 500 to 1000 epochs. We also investigated the Deformable DETR, introduced by Zhu et al [11], as

another possible solution to the problem of slow convergence. This paper introduced a deformable attention module that works with a subset of keys in the key query pair relationship, which allows the Deformable DETR to converge up to ten times faster than a traditional DETR.

Using the information from the previous papers we then trained the Conditional DETR and Deformable DETR models for 50 epochs. These models were able to make predictions with higher confidence, but our quantitative metrics were still somewhat poor. We then chose the Conditional DETR model because of its better performance and trained it for 200 epochs, which provided somewhat comparable results to our other models. We measured the recall, precision, mean absolute precision for model comparison as discussed in our evaluation metrics section.

Con-DETR	mAP 50-90	mAP 50	Precision	Recall
50 epoch	0.083	0.117	0.682	0.109
200 epoch	0.234	0.301	0.534	0.238
Def-DETR	mAP 50-90	mAP 50	Precision	Recall
50 epoch	0.006	0.008	0.642	0.022

TABLE II
EVALUATION METRICS OF CONDITIONAL AND DEFORMABLE DETR MODELS

These results reveal an interesting pattern. Specifically, while the mean average precision improves as training increases, the precision actually slightly decreases. This may be due to the fact that although the larger model becomes better at detecting a greater number of objects, it experiences a slight decline in the accuracy of its classifications. This is most likely caused by the smaller models being able to learn the larger objects, while the bigger models are able to detect the smaller objects without being able to classify them as well.

These results are somewhat worse compared to the other models. This has many potential causes, but we believe the main cause is underfitting. Due to the complexity of transformer models, even using DETR models designed to converge faster, these models tend to take an extended number of epochs to fully converge. This can be seen in figure 9 as we notice that the loss shrinks but still appears to be able to get smaller without flattening out.

Another potential cause of our poor results was the problem with stability of the bounding boxes in the data. Qualitative results from both the models in this section and other sections noted how difficult the task of object detection was given our data set. There were multiple instances where even us humans were unable to distinguish some of the smaller classes from the background. These small classes included things like broken glass, or cigarettes, or pop tabs. Using this knowledge of how hard this task was for these tiny classes, we decided to retrain our best preforming model on a variation of our original dataset with these smaller classes removed. Before the removal the average size of our bounding boxes was 2.7 percent of our total 416x416 image, whereas after the removal the average size grew to 3.6 percent of the total image.

After this change to our dataset we retrained the Conditional DETR running for 200 epochs, and found a somewhat

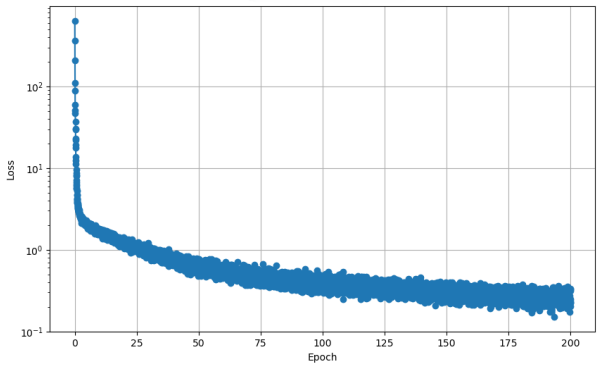

Fig. 9. Test Loss of Conditional DETR over 200 Epochs

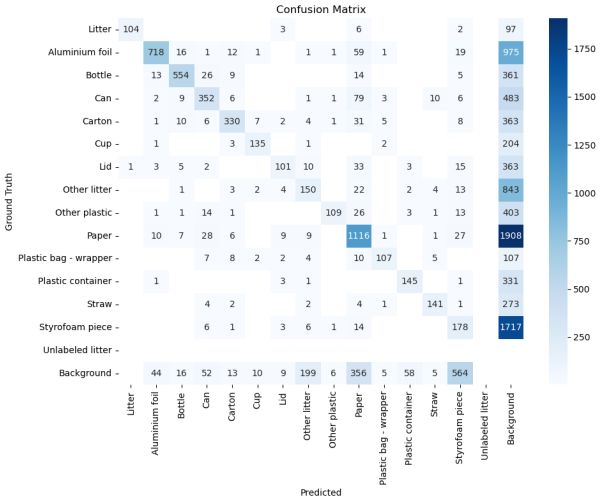

Fig. 10. Confusion Matrix of Conditional DETR trained on cleaned dataset for 200 epochs

significant improvement in our results. We were able to raise our mAP 50-90 to a score of 0.260 and our mAP 50 to a score of 0.337. We were also able to improve our precision to a score of 0.612 and our recall to a score of 0.285. This was a significant improvement when compared too the same model trained on the non augmented dataset. The full comparison can be seen in table III and the confusion matrix for this model can be seen in figure 10.

Dataset	mAP 50-90	mAP 50	Precision	Recall
Original	0.234	0.301	0.534	0.238
Cleaned	**0.260**	**0.337**	**0.612**	**0.285**

TABLE III
EVALUATION METRICS OF CONDITIONAL DETR MODELS TRAINED OVER 200 EPOCHS ON BOTH THE ORIGINAL AND CLEANED DATASET

Due to our original idea of using our models in a real-time detection environment, we created a script to load our best Conditional DETR model and run inference in real time. This was done using a NVIDIA GeForce 3070 Ti and we were able to make predictions at around 13 frames per second. Qualitative results from testing this show that our

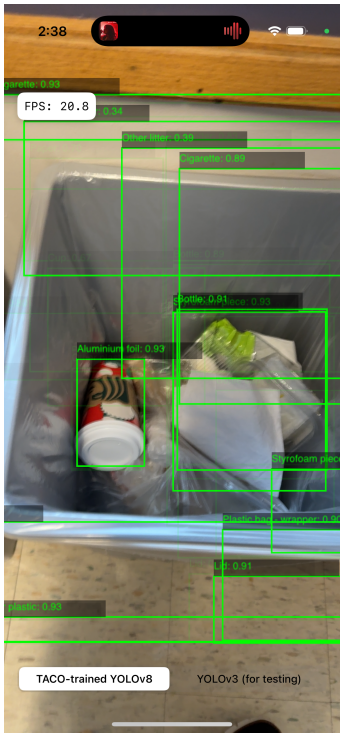

Fig. 11. iOS app screenshot when tested on iPhone 16 Pro

model is able to predict easy to find trash objects with high levels of confidence, but often finds other random objects and misidentifies them as trash.

Overall, the performance of our DETR based models was subpar. We were able to get the models to successfully train, predict, and run in a real time environment. However, our DETR of family models did preform worse than both the YOLO and R-CNN models.

V. Bonus Feature

As a bonus feature, an iOS app for object classification was developed. The app utilizes the trained model from the YOLOv8 section to display annotations on images from the rear camera of the user's iPhone. Tested on an iPhone 16 Pro, the app was able to run inference at an average of 20 FPS as seen in Figure 11. The model was able to detect objects within a trash bin, but struggled with oversensitivity and mislabeling, indicating possible overfitting to the TACO dataset and its noisy backgrounds. We think that a model with masked data could better generalize to diverse backgrounds.

VI. Member Contribution

A. Ryan

I was tasked with the GroundingDINO section of the project, as well as the entirety of the iOS app. I had initially planned on *training* a custom GroundingDINO model with the TACO dataset as I had remembered using the model over the summer for my robotics research, seeing that releasing the training code for the model on the to-dos list of the official repository. However, the training code remains unreleased so I was only able to "optimize" the model for this use case via hyperparameter tuning. Our dataset was separated by annotation, not by image, so I had to write a custom parser and dataloder that would return each individual image with all of its associated annotations for the evaluation script. I additionally needed to implement logic to load the data and perform inference in an efficient manner, as inference was already a slow process even on workstation hardware.

For the iOS app, I needed to write logic to handle launching a camera capture session, which required me to get up to speed on Apple's previous UI framework UIKit, as their newer one that I *do* know, SwiftUI, does not yet have support for camera capture. I had to convert the trained YOLOv8 model to Apple's CoreML format, and learn the API for performing inference and retrieving the results from the converted model. I then had to transform the normalized GroundingDINO bounding box position, height, and width values to the image coordinate system, and then to the camera coordinate system so they could be displayed properly on the screen (without the conversion to the camera coordinate system, the boxes' positions would be mirrored on the horizontal axis).

I feel that even if our detection models didn't have the best performance, we were still able to successfully evaluate their usability for this task, which is what we ultimately set out to do. I gained insight as to how VLMs like GroundingDINO operate, and got hands-on experience with real datasets and creating the tools that are necessary for loading and evaluating them. I was able to learn about choosing correct hyperparameters, prompt engineering for language models, and evaluation methods for model outputs. With regards to the GroundingDINO model specifically, I spent a great deal of time attempting to troubleshoot my evaluation code before coming to the realization that it was simply making such incredibly inaccurate predictions.

B. Robert

I was responsible for researching, training, and evaluating the Faster R-CNN model with a ResNet-50 backbone, as well as writing the conclusion section of this paper. Initially, I considered using ResNet-50 alone because its training on the ImageNet dataset suggested that it would generalize good to our dataset. However, upon further investigation, I realized that ResNet-50 is fundamentally an image classification model that assigns a single label to an entire image. Our dataset, on the other hand contains multiple objects of interest within a single image, each of which needs to be identified and localized. Therefore, I integrated the ResNet-50 backbone with a Faster R-CNN framework, which can generate region proposals and classify each potential object instance. This combination allowed the model to detect and localize multiple items within a single image.

Working on this project deepened my understanding of the practical challenges in computer vision tasks, and helped me learn how to develop a machine learning workflow for a computer vision task. While previous coursework covered

theoretical ideas, this hands-on experience required me to properly load and pre-process a custom dataset and ensure it matched the model's expected input formats. I also learned about choosing appropriate hyperparameters, optimizers, and learning rate schedulers. It often seemed the model's performance would not improve even with different hyperparameter tuning, which prompted me to analyze the results thoroughly and devise innovative solutions such as customizing the sampling strategy to emphasize images with small objects in each batch, and introducing a more robust loss function that penalized missed detections of these difficult cases. These modifications ultimately enhances the model's precision and robustness in detecting small, hard to find objects in cluttered scenes.

C. Isaac

I worked on the DETR section of our project. We originally only came in with the idea to compare some sort of transformer-based approach to our other models, so I was given the choice with where I wanted to take this side of our project. I chose the DETR family of models because it seemed like the most well-used and documented style of models for transformer-based object detection. I was also responsible for finding alternative DETR models when the originally chosen models failed to work. In addition, I introduced the idea of trash detection as a project idea to our group.

I learned much valuable information during this project. I had never done any work with image datasets prior to this project, and I ended up originally struggling to format the data in a usable way for my model training. I ended up taking around a week to try to manually convert YOLO to COCO format before I realized I could simply re-download the dataset. I also had a few issues choosing the correct framework to load and fine-tune these models and ended up trying many architectures before finding one that worked. I also struggled with debugging my model as I was unaware of how severe under-fitting a transformer model affects its performance. However, despite all these setbacks, I have learned invaluable information about DETRs as an architecture and the workflow of a computer vision project, and I am now much more confident with my knowledge of computer vision tools and how to solve problems using computer vision pipelines.

D. Jerome

I worked on the YOLO model section of our project, where I was responsible for training, testing, and evaluating the model. Additionally, I identified and selected the dataset we ultimately used. Through this experience, I gained a deeper understanding of YOLO models, how they work, and how to train them, which is widely regarded as one of the most popular approaches in object detection. I also learned valuable insights about proper data formatting and its critical role in achieving accurate results. Lastly, I learned about dataset manipulation and the effects it has on a model's performance when removing the small object annotations.

VII. Conclusion

A. Summary of Work

In this project, we investigated the application of computer vision and machine learning techniques for waste classification into 18 different classes. We trained and evaluated four different models, GrondingDino, Fine-Tuned DETR, YOLOv8, and ResNet, using the TACO dataset from Kaggle, which contains images of waste in real-world settings. Our evaluation focused on standard object detection metrics, including precision, recall, mAP@50 and mAP@50-95, to provide a comprehensive comparison of each model's performance. Through various experiments, including adjustments to training epochs and modifications to the dataset annotations, We analyzed the strengths and weaknesses of each model in detecting and localizing waste objects of varying sizes, complexities, and occlusions. To conclude our work we created an app that utilizes the YOLOv8x model we trained to detect trash on a phone to see how it performs on an edge device.

B. Key Findings

Our findings revealed that YOLOv8 performed the best in terms of precision, recall mAP50, and mAP50-95 as shown in Table IV, with YOLOv8 achieving notable speed advantages, making it suitable for real-time applications. Over all of the models, we observed that smaller objects like bottle caps and cigarettes were more challenging to detect, particularly in images with real-world clutter, highlighting the limitations of the dataset's resolution and scale.

Evaluation Metric	YOLO	DETR	ResNet	GroundingDino
Precision	**0.777**	0.612	0.503	N/A
Recall	**0.398**	0.285	0.379	N/A
mAP50	**0.491**	0.337	0.352	N/A
mAP50-95	**0.403**	0.260	0.297	N/A
FPS	200	13	10	2

TABLE IV
EVALUATION METRIC SCORES FOR EACH MODEL.

C. Limitations and Future Work

While our experiments provided valuable insights, there were several limitations to our approach. The TACO dataset's inherent challenges such as small object representation and real-world noise impacted model performance, particularly for smaller classes. The computational constraints additionally limited the exploration of more extensive hyperparameter tuning. Future work could address these limitations by incorporating datasets with higher-resolution images and more balanced class distributions. Exploring advanced techniques such as multi-modal learning or edge computing for real-time deployment could further enhances system performance. Expanding the study to include additional datasets and real-world testing environments would provide a more holistic understanding of the models' practical applicability in waste management systems.

Similarly, we were also somewhat limited by computational power. We would have loved to run the base DETR and RT-DETR models for 500, 750, and 1000 epochs respectively but

given our hardware and time constraints, it was not feasible for this project.

D. Code and Models

1) DETR: The code for fine-tuning the DETR models and generating evaluation metrics are available at this GitHub link. Also, all the trained models are available for download here and can be run using the HuggingFace transformers library.

E. YOLOv8

Jerome has created a zip file that he has submitted alongside his report submission. This contains the original TACO dataset training, modified TACO dataset training, and the new test dataset.

1) GroundingDINO YOLO iOS App: The code for the GroundingDINO model evaluation as well as the iOS app for the YOLOv8 model are available here. Note that it was not possible to include the CoreML .mlpackage files for the iOS app in the repository as GitHub limits individual files to 100 MB.

2) FasterRCNN Resnet: The code for training this model can be found here

References

[1] W. Lu and J. Chen, "Computer vision for solid waste sorting: A critical review of academic research," *Waste Management*, vol. 142, pp. 29–43, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0956053X22000678

[2] N. Narayanswamy, A. Rajak, and S. Hasan, "Development of computer vision algorithms for multi-class waste segregation and their analysis," *Emerging Science Journal*, vol. 6, pp. 631–646, 04 2022.

[3] R. Wu, X. Liu, T. Zhang, J. Xia, J. Li, M. Zhu, and G. Gu, "An efficient multi-label classification-based municipal waste image identification," *Processes*, vol. 12, no. 6, 2024. [Online]. Available: https://www.mdpi.com/2227-9717/12/6/1075

[4] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," 2024. [Online]. Available: https://arxiv.org/abs/2303.05499

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.

[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 213–229.

[9] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen, "Detrs beat yolos on real-time object detection," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 16 965–16 974.

[10] D. Meng, X. Chen, Z. Fan, G. Zeng, H. Li, Y. Yuan, L. Sun, and J. Wang, "Conditional detr for fast training convergence," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 3651–3660.

[11] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: deformable transformers for end-to-end object detection," *CoRR*, vol. abs/2010.04159, 2020. [Online]. Available: https://arxiv.org/abs/2010.04159