# Scaling Part Models: Challenges and Solutions for Robustness on Large Datasets

*Nabeel Hingun*
*David A. Wagner, Ed.*

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-153
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-153.html

May 12, 2023

Acknowledgement

I would like to thank Professor David Wagner for his guidance and support throughout my academic journey. His insightful feedback has been instrumental in my research.
I would also like to extend my appreciation to Professor Dawn Song for her help as a second reader.
I would like to acknowledge the invaluable mentorship and support provided by Chawin Sitawarin. His guidance and countless hours of availability have been instrumental in my academic success.
Lastly, I would like to thank my family and friends for their unwavering love and support throughout my academic journey.

# Scaling Part Models: Challenges and Solutions for Robustness on Large Datasets

by Nabeel Hingun

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

**Committee:**

[illegible]

Professor David Wagner
Research Advisor

5/9/2023

(Date)

* * * * * * *

Dawn Song

Professor Dawn Song
Second Reader

5/10/23

(Date)

Abstract

# Scaling Part Models: Challenges and Solutions for Robustness on Large Datasets

by

Nabeel Hingun

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor David Wagner, Chair

Part models have been shown to be an effective way to increase the robustness of deep learning models to adversarial examples. While part models have successfully been applied to small datasets like PartImageNet, obtaining the necessary segmentation labels can be expensive and time-consuming. In order to scale part models to larger datasets, it is crucial to find ways to obtain cheaper labels. In this work, we explore some of the challenges that may arise when scaling up part models. First, we investigate ways to reduce labeling costs by using part bounding box labels instead of segmentation masks, while still providing additional supervision to models. Second, we evaluate the performance of part-based models on a more diverse and larger dataset. Our work provides valuable insights into the key challenges that need to be addressed in order to scale up part models successfully and achieve adversarial robustness on a larger scale. The code is publicly available at https://github.com/nab-126/adv-part-based-models.

# Contents

# List of Figures

# List of Tables

## Acknowledgments

I would like to thank Professor David Wagner for his guidance and support throughout my academic journey. His insightful feedback has been instrumental in my research.

I would also like to extend my appreciation to Professor Dawn Song for her help as a second reader.

I would like to acknowledge the invaluable mentorship and support provided by Chawin Sitawarin. His guidance and countless hours of availability have been instrumental in my academic success.

Lastly, I would like to thank my family and friends for their unwavering love and support throughout my academic journey.

# Chapter 1

# Background

## 1.1 Adversarial Robustness

Machine learning models have achieved remarkable success in a variety of applications, including image classification, speech recognition, and natural language processing. However, these models are susceptible to adversarial examples, which are inputs that are specifically crafted to deceive machine learning models. These adversarial examples are generated by adding small, imperceptible perturbations to legitimate inputs [12] and can cause the model to make incorrect predictions with high confidence. Such attacks pose a significant threat to the security and reliability of machine learning systems, particularly in safety-critical applications such as self-driving cars and medical diagnosis.

There has been significant research in improving the adversarial robustness of machine learning models and their ability to classify adversarial examples correctly. One such approach, called adversarial training, is to train models on adversarial examples during the training process [9] to help the model learn distinguish between benign and adversarial inputs. In fact, adversarial training is the baseline and state-of-the-art approach to adversarial robustness and has led to significant improvements in this area. Yet, it has been observed that the benefits of adversarial training tend to plateau as the number of adversarial examples used during training increases [3]. Achieving linear improvements in adversarial robustness necessitates exponentially more computational resources, which can make this approach prohibitively expensive. This computational cost increases further with larger models and larger datasets.

In light of the increasing costs associated with adversarial training, recent research [11] has explored alternative approaches to achieving robustness through the use of part-based models. The work demonstrates that such models can indeed achieve adversarial robustness, providing a cheaper paradigm to adversarial training.

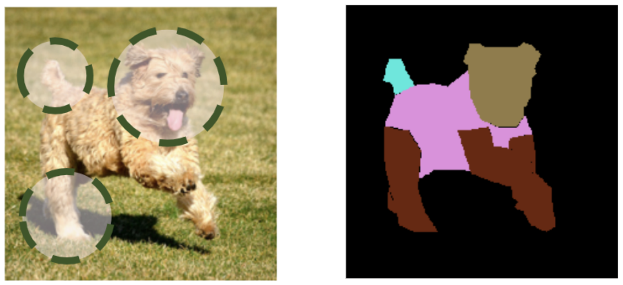

Figure 1.1: Example of a Part Segmentation Mask

## 1.2 Part-Models

Part-based models are usually used to improve object recognition in situations when objects can vary in appearance due to changes in lighting, viewpoint, or occlusion. The basic idea behind part-based models is to break down an object into its constituent parts which are then used in downstream tasks.

Sitawarin et al. [11] show that the combination of the part-based model architecture and adversarial training improves adversarial robustness. The authors introduce four part-based models – *Downsampled*, *Bounding-Box*, *Two-Headed*, and *Pixel* – which all perform equally well in terms of adversarial robustness. For a given image, these models are trained to predict both its label and part segmentation mask. Part segmentation masks are a type of mask that provide more detailed annotation of an object by labeling different parts separately, as opposed to having a single mask for the entire object. As a result, they are able to provide a more fine-grained segmentation of an image. For instance, the part segmentation mask depicted in Fig. 1.1 highlights the head, body, legs, and tail of a dog. The authors note that their part-based models can be used with any segmenter architecture. In their experiments, they use the DeepLabv3+ segmenter [2] with a ResNet-50 backbone [5].

The *Downsampled* part-based model is a two-stage classifier, i.e, a segmentation model first takes in an image and generates part segmentation masks, which are then passed to a classifier head to predict the class label of the input. As shown in Fig. 1.2, the model reduces the size of the predicted segmentation masks before passing them through a small classifier head for the final image classification. With this sequential architecture, the classifier head only sees the predicted segmentation masks, not the original images. In contrast, the image classifier head from the Two-Headed model sees the dense representation from the bottleneck layer of DeepLabv3+. Its segmentation head uses that same dense representation for part mask predictions. Please see Fig. 1.3 for an illustration of this architecture.

While these part-based models have shown promising results in improving adversarial robustness, they still rely on the use of expensive segmentation masks as a form of training supervision. The high cost of obtaining such fine-grained labels limits the adoption and practical applicability of these models. To address this issue, we explore the possibility

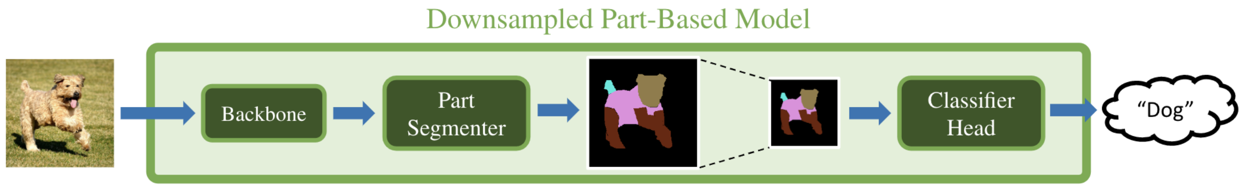

Figure 1.2: Downsampled DeepLabv3+ Architecture

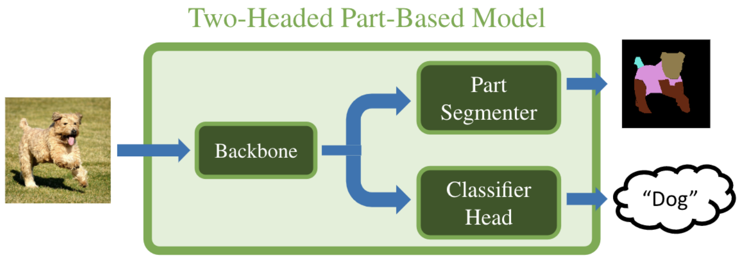

Figure 1.3: Two-Headed DeepLabv3+ Architecture

of using bounding box labels as a cheaper alternative to segmentation masks. We also investigate the performance of state-of-the-art models for object detection, namely DINO [13], for generating bounding box labels.

## 1.3 DINO and Mask DINO

DINO (DETR with Improved deNoising anchOr) [13] is a transformer-based model used for object detection. The model takes in an image and predicts a bounding box for every object within that image. Overall, the model learns to reason about all objects in an image by attending to these different objects and encoding them into a high-dimensional vector representations.

DINO's architecture consists of a backbone, a multi-layer encoder, a multi-layer decoder, and a feed-forward network. First, the backbone extracts feature representations of input images. These representations are then processed by the encoder. The decoder layers attend to the encoder outputs with learnt position embeddings. These position embeddings, called object queries, are formulated as dynamic anchor boxes and are refined step-by-step across decoder layers. Finally, the decoder output is passed through the feed-forward network to predict the normalized center coordinates and class label of each bounding box. DINO predicts a fixed-size set of bounding boxes which is usually much larger than the actual number of objects of interest in an image.

There are multiple reasons we choose to use DINO. First, DINO makes several improvements over previous transformer-based object detectors [1] in terms of training efficiency and query initialization. Second, DINO achieves state-of-the-art performance on the COCO [7] public benchmarks. More importantly, DINO eliminates the need of hand-crafted algorithms like non-maximum suppression or anchor generation and makes the detection pipeline end-to-end differentiable. This ensures we can perform adversarial training. Finally, DINO also uses a ResNet-50 backbone and hence uses the same dense representations from images as the part-models in previous work [11].

Li et al. [6] recognize the lack of transformer-based models for both detection and segmentation tasks. The existing ones have yet to achieve the same level of performance on these tasks as CNN-based models using such unified frameworks. Hence, they propose a unified approach to leverage the shared objective of localizing objects in an image and learning more useful image representations. In particular, they introduce Mask DINO, a segmentation model that extends DINO with an additional mask prediction head. Mask DINO uses the same position embeddings from DINO to predict segmentation masks, while retaining the state-of-the-art performance, training efficiency, and end-to-end differentiability of DINO.

# Chapter 2

# Object Detection

Recent work [11] has proposed a novel approach to improving adversarial robustness with part-models. This suggests a new direction for work on robustness, based on richer supervision using segmentation masks. However, the main limitation is the high cost of obtaining segmentation labels. To address this limitation, we explore the use of bounding boxes as a cheaper form of additional supervision. In this context, we assume the model to be trained on a labeled set of images where each image is associated with a class and bounding boxes that indicate the location of the objects within the image. Please refer to Fig. 2.1 for an illustration of this part detection architecture. In our work, we compare the performance of detection models to that of segmentation models in terms of clean and robust accuracy (see Table 2.1 for a summary of our results).

## 2.1 Dataset Overview

We use PartImageNet [4] as a benchmark for evaluating the adversarial robustness of our part-models. PartImageNet is a large and high-quality dataset consisting of approximately 24,000 images from 158 classes of ImageNet, grouped into 11 supercategories. We reshuffle the dataset into a 0.8/0.1/0.1 train/val/test split and hold out the test set for evaluation. Since the PartImageNet dataset follows the COCO-style annotation format, each part segmentation mask has an associated bounding box. It is not necessary for us to convert ground truth segmentation masks to ground truth bounding boxes.

| Model | Clean Acc. | Robust Acc. |
|---|---|---|
| Segmentation Model | **84.1** | **40.5** |
| Detection Model | 83.4 | 38.7 |

Table 2.1: Overview Comparison of our best Part Segmentation and best Part Detection Models on PartImageNet

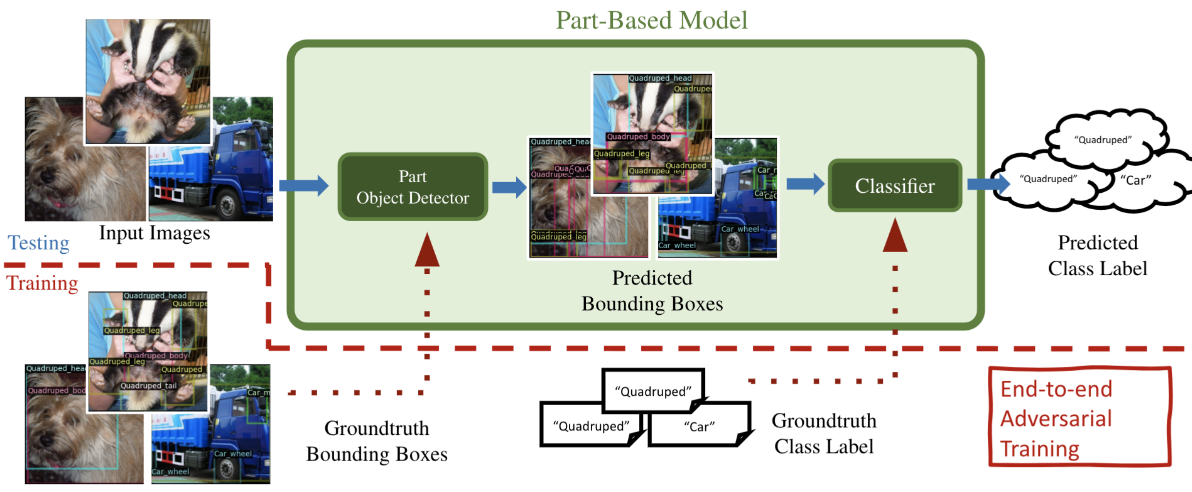

Figure 2.1: Architecture used for Part Bounding Box Classification

## 2.2 Model Architecture

Sitawarin et al., explored using bounding boxes as an alternative labeling method, instead of part segmentation masks. In their experiments, they used the DeepLabv3+ segmentation model on part box segmentation masks (see row (b) of Fig. 2.4 for examples) and measured the model's accuracy in predicting the class of the image. Even though the part box segmentation masks are less fine-grained than the original part segmentation masks, they found that bounding box labels were nearly as effective as segmentation masks, achieving 84.1% clean accuracy and 39.7% adversarial accuracy on PartImageNet. Building on this line of work, we further investigate the use of bounding boxes as a cheaper form of additional supervision. Specifically, instead of using a segmentation model on bounding box labels, we use DINO, an architecture specialized for object detection. Our model is trained on images with bounding box labels for each part.

Fig. 2.2 illustrates the two-stage model architecture we use for classification. An image is first fed to DINO which outputs bounding boxes with normalized center coordinates (box center x, box center y, box width, box height) concatenated with their respective class logits, resulting in a feature vector for each detection. These feature vectors are then passed through a fully connected neural network with two hidden layers and a final Softmax output layer to predicts a probability distribution over classes. We refer to this model as *DINO sequential*.

With around 47 million trainable parameters, DINO is a much larger model than Deeplabv3+ (around 27 million trainable parameters). To ensure that any increase in model robustness comes from the additional supervision signal and not from the model size, we also implement a two-headed DINO model (see Fig. 2.3). Using this architecture, the ResNet50 backbone processes the input image and extracts its feature maps. These feature maps are then fed to the object detection head and a classification head. The object detection head follows

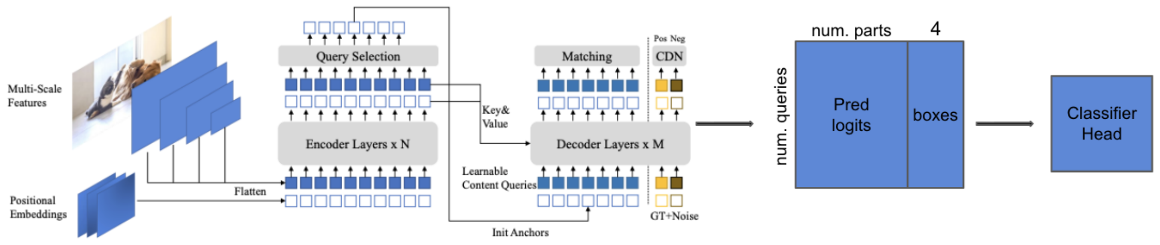

Figure 2.2: The DINO sequential architecture: the predicted bounding boxes (part class logits and normalized coordinates) are used by the classifier head for image classification

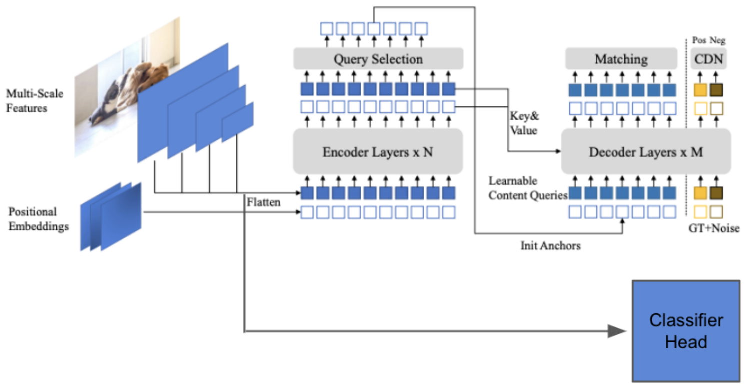

Figure 2.3: The DINO multi-head architecture: the DINO head (top) is used for Part Bounding Box Prediction and the classification head (bottom) is used for Image Classification. Both heads used the same image feature representations extracted by the backbone

the standard DINO architecture to predict bounding boxes while the classification head first reduces the spatial dimensions of the feature maps with average pooling then applies a fully connected layer to obtain the final classification logits. The entire model is trained end-to-end using a joint loss that combines the detection and classification objectives. With the two-headed model, the detection head can be discarded after training and we will hopefully be left with a more robust model. We refer to this model as *DINO two-headed* in our experiments.

## 2.3 Metrics

In our experiments, we utilize standard evaluation metrics to assess the performance of our models. For the classification task, we use accuracy, which measures the proportion of correctly classified samples.

For the segmentation task, we use pixel-wise accuracy which measures the proportion of correctly classified pixels in the image. Specifically, for every pixel of an image, we obtain its predicted class label by computing the argmax of the pixel's class logits. Then, we compare the predicted class label to its corresponding pixel in the ground truth segmentation mask. If both pixels are classified as the same label, we consider it as a true positive. The pixel-wise accuracy is then calculated by dividing the total number of correct predictions by the total number of pixels in the image.

For the object detection task, we use Mean Average Precision (MAP) which is calculated as the average of the precision values at different recall levels. Here, precision is the ratio of true positive detections to the total number of detections. We use two variants of MAP: mAP@0.5 and mAP@0.5:0.95. mAP@0.5 measures the average precision of the model at a fixed Intersection over Union (IoU) threshold of 0.5. mAP@0.5:0.95, on the other hand, measures the average precision of the model at a range of IoU thresholds from 0.5 to 0.95.

Overall, the accuracy metric provides a straightforward measure of classification performance while the pixel-wise accuracy and MAP metrics enable us to evaluate the quality of the predicted segmentation masks and bounding boxes, respectively, in comparison to the ground truth annotations.

## 2.4 Converting between segmentation masks and bounding boxes

Object detectors and image segmenters use different evaluation metrics. Hence, to compare their performance, we need either need to convert the predicted segmentation masks to bounding boxes or convert the predicted bounding boxes to segmentation masks.

To derive bounding boxes from predicted segmentation masks, we use a method to generate a bounding box for each connected component in the segmentation mask. Here, a connected component is defined as a collection of pixels that are connected to each other and assigned the same label. Specifically, we first produce the predicted segmentation mask by using the softmax function to produce a probability map for each pixel and subsequently assigning each pixel to the label with the highest probability. For each connected component in this predicted segmentation mask, we compute the minimum and maximum pixel coordinates within the component, which correspond to the top-left and bottom-right corners of the bounding box, respectively. Finally, we transform these corner coordinates normalized center coordinates for the bounding box. The confidence score for each bounding box is calculated as the average of the probability values in the predicted segmentation mask that correspond to the connected component. To prevent the creation of bounding boxes for small connected components of pixels, we exclude those with an area less than 1% of the total image area. Row (c) and (d) from Fig. 2.4 illustrates this conversion of segmentation masks to bounding boxes.

The process of generating segmentation masks from bounding boxes involves creating

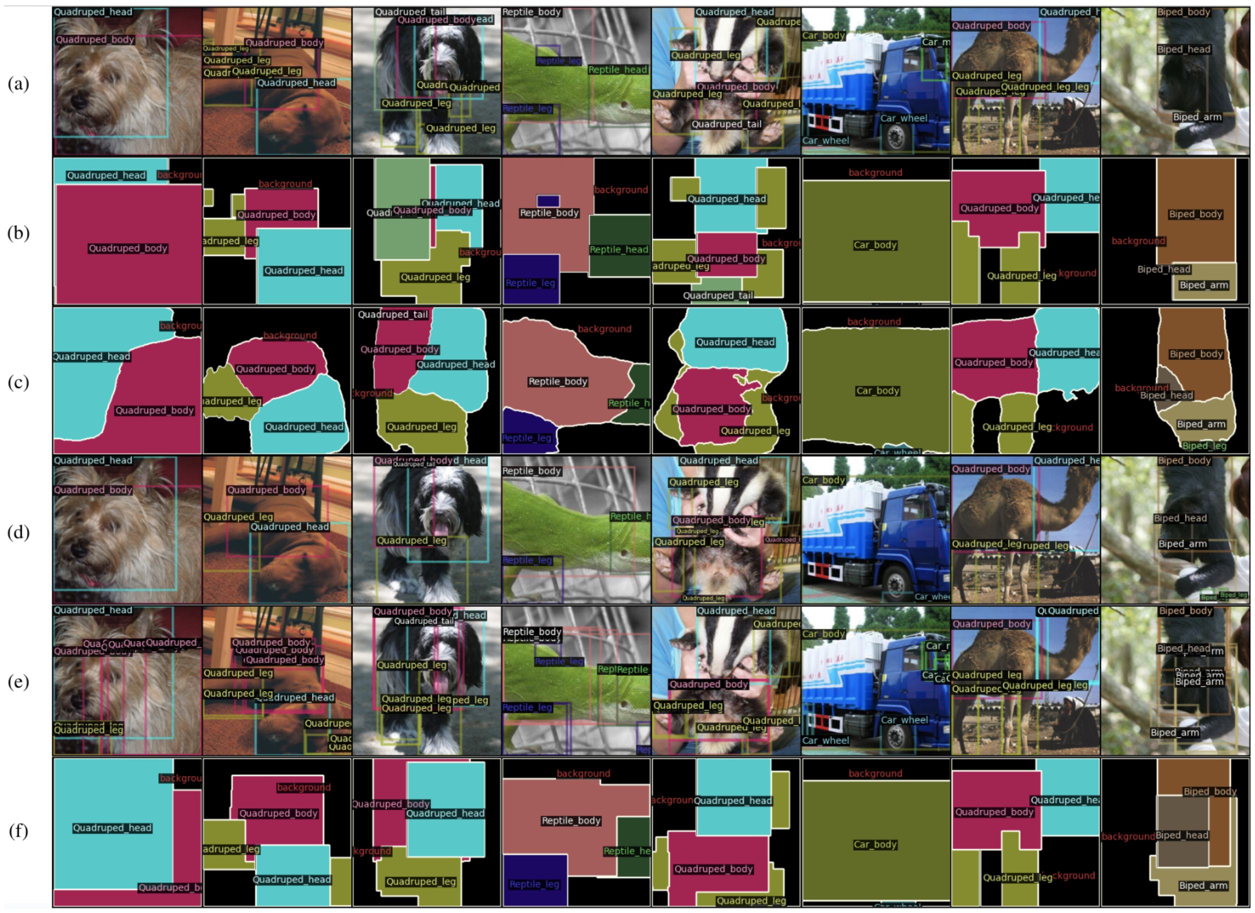

Figure 2.4: Deeplabv3+ and DINO Example Predictions
(a) Original images with Ground Truth bounding boxes
(b) Ground Truth segmentation masks
(c) Deeplabv3+ Predicted segmentation masks
(d) Deeplabv3+ Predicted segmentation masks converted to bounding boxes
(e) DINO Predicted bounding boxes (top 10 only)
(f) DINO Predicted bounding boxes converted to segmentation masks

a binary mask for each object detected within an image. First, bounding boxes with low confidence scores are filtered out based on a confidence threshold. For each pixel in the image, the corresponding segmentation label is determined by selecting the bounding box with the maximum confidence score that covers the pixel. Since the object detector does not predict the background class, the absence of a bounding box covering a particular pixel indicates the presence of the background class. To determine the optimal confidence threshold, we test the different values in the range 0.1-0.5. Using a high threshold like 0.5 results in the model mostly predicting the background class since few bounding boxes have a confidence score higher than that. Conversely, using a low threshold predicts too few instances of the background class. We settle on a threshold of 0.2 in our experiments. Please see row (e) and (f) from Fig. 2.4 for some examples of converting bounding boxes to segmentation masks.

## 2.5 Experiments

In all of our experiments, we train our models with adversarial training to encourage them to be robust to $L_\infty$ perturbations. To stay consistent with previous work [11] and to speed up adversarial training, our models are first pretrained for 50 epochs on clean samples and are then adversarially trained with 10-step PGD [9] with $\epsilon = \frac{8}{255}$ for another 50 epochs. For evaluation, we use 100-step PGD also with $\epsilon = \frac{8}{255}$ on the held-out test set.

### Segmentation vs Detection

We present a comparative analysis of segmentation models and detection models, using only segmentation or detection tasks for evaluation and excluding the classification task. For a fair comparison of models using the same annotations, we train the segmentation model on bounding box segmentation labels (see second row from Fig. 2.4).

By converting between segmentation masks and bounding boxes, we can evaluate the performance of our object detector using the same metrics as the segmentation model. This allows us to make a fair comparison between the two models. We train a Deeplabv3+ model as our segmenter on part box segmentions and train a DINO model as our object detector on part bounding box annotations (see Fig. 2.4).

We experiment with different hyperparameters for DINO by varying the number of queries and the number of encoder/decoder layers of the transformer. Specifically, we consider 100 queries and 900 queries (the default for DINO), as well as a smaller transformer with only one encoder/decoder layer, instead of the default 6 layers. Our experimental results demonstrate that both the segmentation and detection models achieve high performance on their respective tasks on clean samples (as shown in Table 2.2). For instance, the pretrained Deeplabv3+ model performs better than the pretrained DINO model in terms of pixel-wise accuracy, while the pretrained DINO model outperforms Deeplabv3+ on MAP scores. However, while the robust Deeplabv3+ model still outperforms the robust DINO model on both clean and adversarial samples in terms of clean accuracy, the robust DINO model exhibits low MAP scores comparable to those of Deeplabv3+.

### Detection Part Models

In this experiment, we investigate the robustness of detection part-based models in comparison to segmentation part-based models at the task of classifying images from PartImageNet, as shown in Table 2.3. We evaluate the performance of the DeepLabv3+ *Downsampled* model

| | | | | Pretrained + Clean Images | | | Adv. Trained + Clean Images | | | Adv. Trained + Adv. Images | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Num. Params | Enc/Dec. Layers | Num. Queries | Pixel Acc. | mAP@.5 | mAP@.5:0.95 | Pixel Acc. | mAP@.5 | mAP@.5:0.95 | Pixel Acc. | mAP@.5 | mAP@.5:0.95 |
| Deeplabv3+ | 26.7 | | | **83.8** | 0.147 | 0.312 | **70.1** | 0.076 | **0.175** | **52.5** | 0.023 | 0.055 |
| DINO | 46.6 | 6 | 900 | 82.3 | **0.338** | 0.560 | 46.2 | 0.004 | 0.002 | 46.2 | 0.002 | 0.005 |
| DINO | 46.4 | 6 | 100 | 82.4 | 0.337 | **0.565** | 46.2 | 0.002 | 0.005 | 46.2 | 0.001 | 0.004 |
| DINO | 32.7 | 1 | 900 | 80.4 | 0.270 | 0.454 | 56.6 | **0.080** | 0.151 | 46.3 | **0.030** | **0.062** |
| DINO | 32.5 | 1 | 100 | 80.5 | 0.263 | 0.454 | 46.2 | 0.002 | 0.005 | 46.2 | 0.002 | 0.005 |

Table 2.2: Comparison of part segmentation and part detection models, when trained on the part segmentation or part detection task but not on the classification task

| | | | | Pretrained + Clean Images | | | | Adv. Trained + Clean Images | | | | Adv. Trained + Adv. Images | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Num. Params | Enc/Dec. Layers | Num. Queries | Pixel Acc. | mAP@.5 | mAP@.5:0.95 | Acc. | Pixel Acc. | mAP@.5 | mAP@.5:0.95 | Acc. | Pixel Acc. | mAP@.5 | mAP@.5:0.95 | Acc. |
| Deeplabv3+ | 26.8 | | | 82.2 | 0.122 | 0.267 | 98.8 | 71.7 | 0.067 | 0.161 | 84.1 | 54.7 | 0.030 | 0.064 | **40.6** |
| DINO seq. | 49.5 | 6 | 900 | **83.0** | 0.342 | 0.568 | 98.8 | **73.2** | **0.248** | **0.422** | **86.6** | 51.8 | 0.096 | 0.158 | 36.6 |
| DINO seq. | 47.0 | 6 | 100 | 82.7 | 0.335 | 0.564 | 98.7 | 71.1 | 0.227 | 0.391 | 83.2 | 50.4 | 0.082 | 0.143 | 36.5 |
| DINO seq. | 35.4 | 1 | 900 | 79.8 | 0.245 | 0.424 | 98.6 | 69.3 | 0.158 | 0.276 | 83.7 | 51.2 | 0.061 | 0.103 | 38.5 |
| DINO seq. | 32.8 | 1 | 100 | 80.0 | 0.245 | 0.428 | **98.9** | 67.6 | 0.152 | 0.271 | 83.2 | 49.2 | 0.059 | 0.103 | 37.0 |
| DINO two-head | 46.9 | 6 | 900 | 82.6 | 0.342 | 0.567 | 99.0 | 58.8 | 0.166 | 0.277 | 48.2 | 69.6 | 0.169 | 0.309 | 31.9 |
| DINO two-head | 46.7 | 6 | 100 | 82.9 | **0.346** | **0.574** | **98.9** | 59.5 | 0.173 | 0.297 | 55.0 | **69.1** | **0.172** | **0.331** | 32.4 |

Table 2.3: Comparison of part segmentation and part detection models, when trained with a joint classification and segmentation/detection loss

[11], as well as two DINO-based models ;*DINO sequential* and *DINO two-headed.* Even when varying the number of encoder/decoder layers and number of queries for DINO based models, we still observe that the DINO models are outperformed in terms of robustness, as evidenced by their low accuracy on adversarial images.

## 2.6 Conclusion

Our experiments have two main findings:

1. We demonstrate that it is possible to achieve adversarial robustness through object detection. Using part bounding box instead of fine-grained segmentation masks reduces annotation costs while still providing an additional form of supervision to models.

2. DeepLabv3+ based models work better than DINO-based models for robust classification. Our study reveals that DINO is a challenging model to train in an adversarial setting. DINO-based models have comparable performance to DeepLabv3+ based models on clean samples but their performance deteriorate significantly on adversarial samples. One possible explanation to this poor performance when exposed to adversarial training is that DINO is a very large model with a considerable number of hyperparameters to tune. We encourage future work to explore how to improve adversarial training with transformer-based models. Ultimately, DeepLabv3+ based models outperform the DINO-based models, with the same labels and same training data and

without the need to tune any hyperparameters. Hence, DeepLabv3+ based models are a better choice for adversarial robustness.

# Chapter 3

# PACO

To achieve high performance in object recognition tasks, it is crucial to train models on large and diverse datasets with high-quality annotations. Although the PartImageNet dataset is beneficial for evaluating the robustness of part-models, it still suffers from some limitations in terms of size and annotation diversity. As a natural progression towards overcoming these limitations, we employ PACO (Parts and Attributes of Common Objects) [10] to evaluate part-models on a more challenging dataset with common everyday objects (please refer to Table 3.1 for summarized results).

## 3.1 PACO Overview

PACO provides annotations for various vision tasks, including part-segmentation. The dataset is an order of magnitude larger and more diverse than PartImageNet, spanning 75 object categories and 456 object part categories. It consists of 57,643 images from the LVIS image dataset and 26,384 images from the Ego4D video dataset.

| Model | Clean Accuracy | Robust Accuracy |
|---|---|---|
| ResNet50 | 28.3 | 11.6 |
| Downsampled DeepLabv3+ | 46.7 | 17.5 |
| Two-Headed DeepLabv3+ | 46.9 | **18.4** |
| Mask DINO Simple | **52.0** | 17.7 |

Table 3.1: Overview of our best Part Segmentation Models on PACO

## 3.2 Dataset Preparation

Since PACO was originally built for part mask segmentation, we have to create a classification task from the dataset. First, we discard the test set from the Ego4D dataset because its segmentation masks were held out, leaving us with a total of 74,135 images. Then, using the provided annotated bounding boxes, we crop out 310,603 objects from these images. During that process, we discard small objects whose bounding box area is less than 1% of the image, which leaves us with 123,315 objects left for classification.

For each object, we expand its bounding box's width and height by 1.5 times and crop it out of the original images. This ensures that the extracted object includes relevant information about the background. To get the object's respective segmentation mask, we use the same cropping strategy on the full annotated segmentation mask of the image. This ensures that the part-models are not penalized for predicting the masks of other objects if there are multiple objects occurring in the same crop.

There are 19,030 objects from the PACO dataset without part annotations. We decide not to discard these as they represent about 15.4% of all objects. Instead, we create a special mask label for such objects such that they are ignored when computing the loss for the part-model.

Our created dataset consists of 123,315 images partitioned in a 0.8/0.1/0.1 train/val/test split. We provide a visual overview of the dataset by presenting some example images in Fig. 3.1. The distribution of our dataset across classes can be found in Fig. 3.2.

## 3.3 Architecture

We compare the adversarial robustness of part-based segmentation models to a ResNet50 baseline. In particular, we use the existing *Downsampled* and *Two-Headed* part-based models [11] and introduce a *Mask DINO Simple* model.

The *Mask DINO Simple* model is also a part-based classifier using Mask DINO [6] as a segmenter. It takes in an image to predict part segmentation masks and class labels for the input image. The architecture is similar to a two-headed model whereby the classification head does not use the segmentation masks directly but instead maps Mask DINO's query embeddings to class logits.

More specifically, the classification head computes the class logits using a matrix multiplication operation between the query embeddings and a part-to-class matrix. We can define the part-to-class matrix as $M \in \mathbb{R}^{P \times C}$, where $P$ is the number of parts (excluding the background class) and $C$ is the number of classes (also excluding the background class). Each entry in $M$ represents the mapping between a part and a class. We can also define the query embeddings as $Q \in \mathbb{R}^{N \times P}$, where $N$ is the number of queries. Each entry in $Q$ represents the logits of a query for a particular part class.

The class logits for each query can then be computed using a matrix multiplication operation as follows:

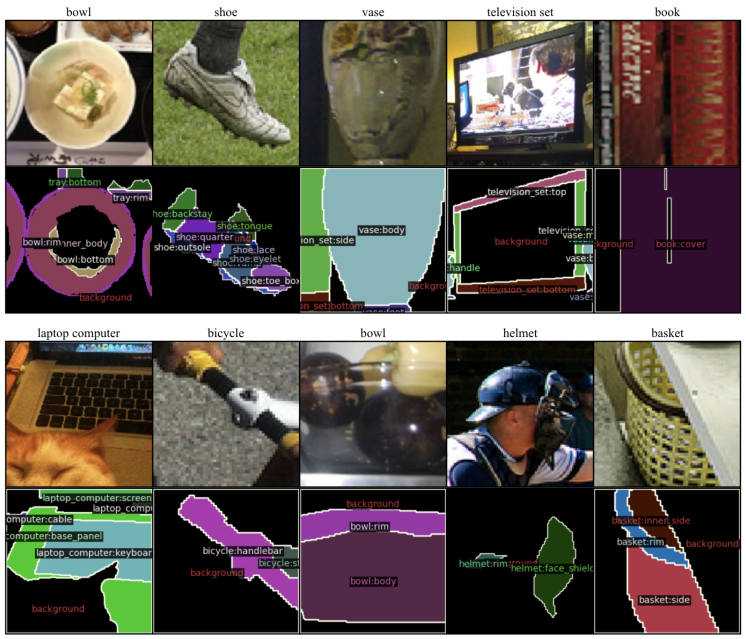

Figure 3.1: Example Images, Segmentation Masks and Class Labels from our created PACO classification dataset

$$B = Q\dot{M} \in \mathbb{R}^{N\times C}$$

This operation applies the mapping between the parts and the classes to the query class logits and produces a matrix of class logits for each query. Finally, the class logits for each query are averaged over the queries to obtain the class logits:

$$c = eB \in \mathbb{R}^{C}$$

where $e = \left[\frac{1}{N} \dots \frac{1}{N}\right]$ and $c$ is the vector of class logits for the input.

## 3.4 Experiments

In our experiments, we evaluate the part-based models on the PACO dataset and present the results in Table 3.4. We observe that all part-based models outperform the ResNet50 baseline in both clean and adversarial accuracy. However, we also notice that adversarial accuracies on PACO are lower than those on Part ImageNet, highlighting the increased diversity and difficulty of the PACO dataset.

Interestingly, we find that the part model utilizing MaskDINO as segmenter achieves similar clean accuracies as the DeepLabv3+-based models. However, the robustness of the

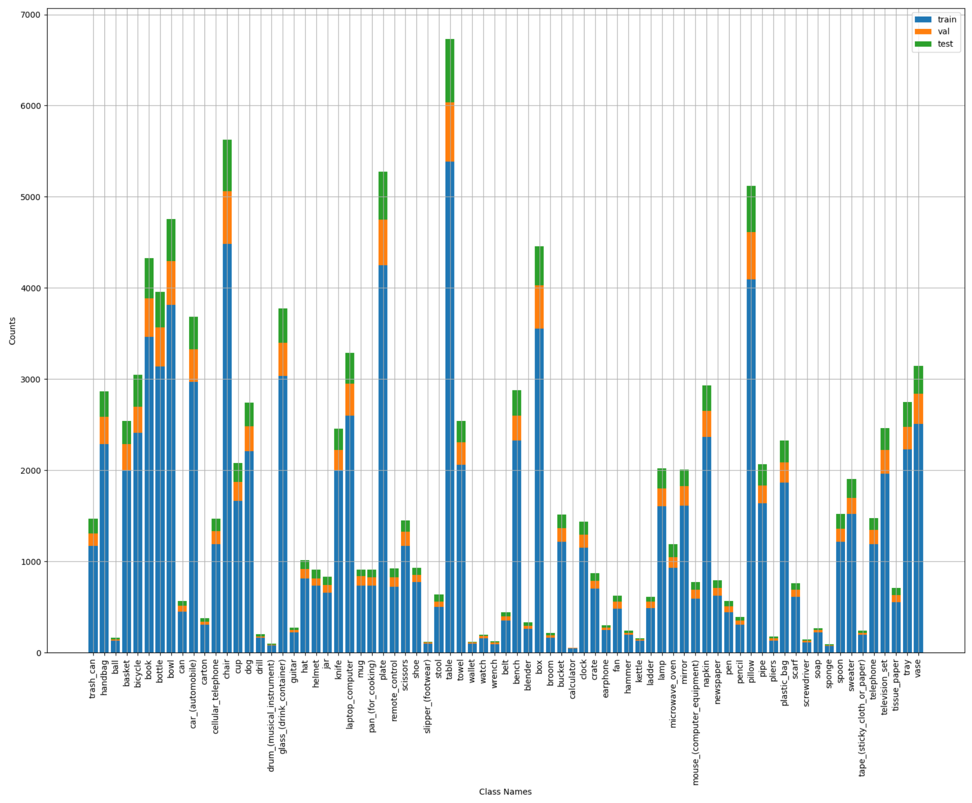

Figure 3.2: Data Distribution for Classification Dataset Created from PACO

MaskDINO part-based model is significantly lower in terms of pixel accuracy and class accuracy on adversarial examples.

To investigate this discrepancy, we conduct additional experiments on the MaskDINO part-based model. First, we examine whether a transformer-based model might require additional PGD steps during training. To do so, we perform an ablation study on the number of PGD steps used during training and evaluation. Specifically, we train a part-based model with 5-step and 10-step PGD and evaluate the model on adversarial examples generated with 5, 10 and 100 PGD steps. Our experiments show that for the Deeplabv3+ model, using more PGD steps during training results in an increase in adversarial robustness ranging from 0.6% to 0.9%, which is similar to the 0.7% to 1.0% observed increase for MaskDINO. Hence, while our findings suggest that additional PGD steps during training can improve adversarial robustness, they do not fully explain the low robustness of the MaskDINO part-based model.

| | | 5 Step PGD Training | | 10 Step PGD Training | | |
|---|---|---|---|---|---|---|
| Model | Num. PGD Eval. Steps | Robust Pixel Acc. | Robust Accuracy | Robust Pixel Acc. | Robust Accuracy | Change in Robust Acc. |
| Deeplabv3+ Two-Headed | 100 | 60.5 | 17.5 | 60.9 | 18.4 | +0.9 |
| | 10 | 60.6 | 18.3 | 61.1 | 19.0 | +0.7 |
| | 5 | 60.7 | 18.7 | 61.2 | 19.4 | +0.6 |
| MaskDINO Simple | 100 | 54.7 | 13.9 | 55.2 | 14.6 | +0.7 |
| | 10 | 55.7 | 15.1 | 56.1 | 16.1 | +1.0 |
| | 5 | 56.3 | 16.1 | 56.6 | 17.0 | +0.9 |

Table 3.2: Comparing the Robustness of models trained with different PGD step sizes against attacks of varying strengths

We also experiment with varying the $c_{\text{seg}}$ hyperparameter during training. This parameter combines the loss of the segmentation task and the loss of the classification task as follows:

$$\text{Loss} = (1 - c_{\text{seg}})\text{Loss}_{\text{classifier}} + c_{\text{seg}}\text{Loss}_{\text{segmenter}}$$

A higher $c_{\text{seg}}$ value places more emphasis on the segmentation task during training. In our experiments, we use $c_{\text{seg}} = 0.5$ for the DeepLabv3+ based models. We demonstrate in Table 3.3 that increasing the value of $c_{\text{seg}}$ for *Mask DINO Simple* leads to improved pixel accuracy for both clean and adversarial images but also results in a decrease in clean and robust class accuracy. Hence, we are not able to match the performance of the Deeplabv3+ model when varying the $c_{\text{seg}}$ hyperparameter.

Finally, we explore some modifications to our *Mask DINO Simple* architecture to improve its performance. In particular, the classification head of the model maps query embeddings relating to object parts in an image to class logits. Since this mapping is hard-coded, we test out a learnable mapping from object parts to classes. Additionally, we experiment with a smaller model by training a MaskDINO part-model with a single decoder layer instead of the default 9 decoder layers since Li et al. [6] have shown that earlier decoder layers in the MaskDINO model have good segmentation performance. By incorporating these modifications, we observe a substantial increase in adversarial robustness for the *Mask DINO Sim* part-model and achieve comparable performance to the DeepLabv3+ part-based model (please see Table 3.4).

We provide visual examples of the predictions from the *Two-Headed* Deeplabv3+ model in Fig. 3.3 and from the *Mask DINO Sim* model in Fig. 3.4.

## 3.5 Conclusion

In our study, we explore the challenges in scaling part models to a larger and more diverse dataset such as PACO. Our experiments reveal that part-models still exhibit improvements

| | | Pretrained + Clean Images | | Adv. Trained + Clean Images | | Adv. Trained + Adv. Images | |
|---|---|---|---|---|---|---|---|
| Model | c_seg | Pixel Acc. | Accuracy | Pixel Acc. | Accuracy | Pixel Acc. | Accuracy |
| Deeplabv3+ Two-Headed | 0.5 | **72.7** | **73.6** | **66.2** | **48.4** | **61.9** | **18.7** |
| MaskDINO | 0.0 | 71.5 | 73.0 | 64.8 | 46.9 | 60.9 | 18.4 |
| MaskDINO | 0.3 | 72.7 | 67.9 | 65.2 | 47.3 | 55.0 | 14.3 |
| MaskDINO | 0.5 | 72.4 | 67.3 | 65.5 | 46.0 | 55.0 | 12.9 |
| MaskDINO | 0.7 | 72.2 | 64.3 | 65.7 | 45.6 | 54.9 | 11.5 |
| MaskDINO | 0.9 | 71.7 | 54.4 | 66.1 | 43.3 | 56.1 | 7.6 |

Table 3.3: Evaluation of MaskDINO Robustness with different hyperparameters combining the segmentation loss and the classification loss

| | Pretrained + Clean Images | | Adv. Trained + Clean Images | | Adv. Trained + Adv. Images | |
|---|---|---|---|---|---|---|
| Model | Pixel Accuracy | Accuracy | Pixel Accuracy | Accuracy | Pixel Accuracy | Accuracy |
| ResNet50 | | 70.6 | | 28.3 | | 11.6 |
| Downsampled DeepLabv3+ | 70.3 | 71.7 | 64.2 | 46.7 | 60.5 | 17.5 |
| Two-Headed DeepLabv3+ | 71.5 | 73.0 | **64.8** | 46.9 | **60.9** | **18.4** |
| Mask DINO Simple | **86.3** | **73.8** | 64.1 | **52.0** | 55.6 | 17.7 |

Table 3.4: Part-based models evaluation on the PACO dataset

in adversarial robustness in comparison to baselines. However, while part-models have reasonable clean accuracy, we encourage future work to increase their adversarial robustness even further. For instance, future work could explore using larger baseline models and larger backbones [8] to further improve the robustness. Finally, the PACO dataset also provides textual object attributes. It would be interesting to explore the use of these attributes as an alternative supervision signal to segmentation masks.

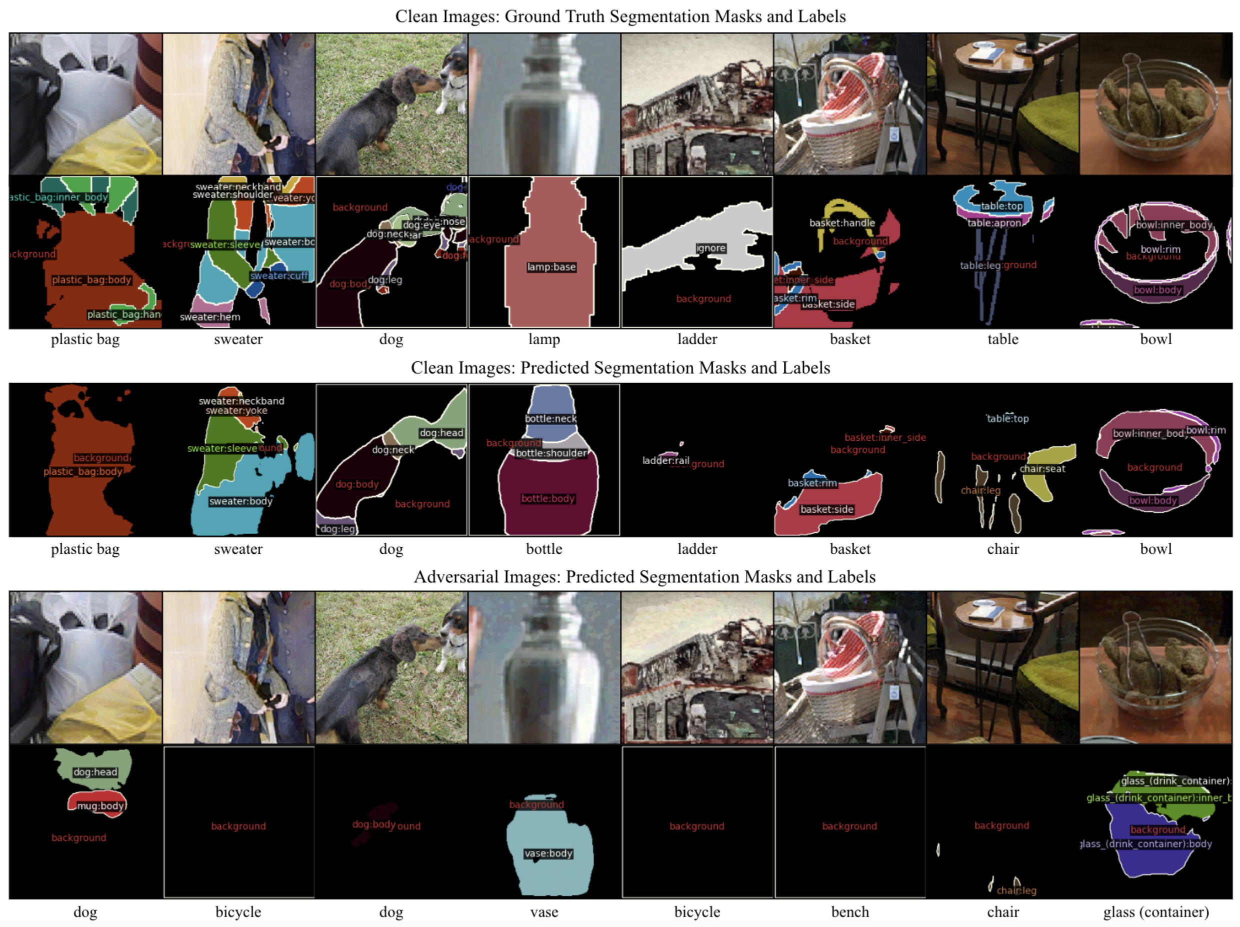
Clean Images: Ground Truth Segmentation Masks and Labels
plastic bag
sweater
dog
lamp
ladder
basket
table
bowl
Clean Images: Predicted Segmentation Masks and Labels
plastic bag
sweater
dog
bottle
ladder
basket
chair
bowl
Adversarial Images: Predicted Segmentation Masks and Labels
dog
bicycle
dog
vase
bicycle
bench
chair
glass (container)

Figure 3.3: Predictions for Two-Headed Part-based Model with DeepLabv3+

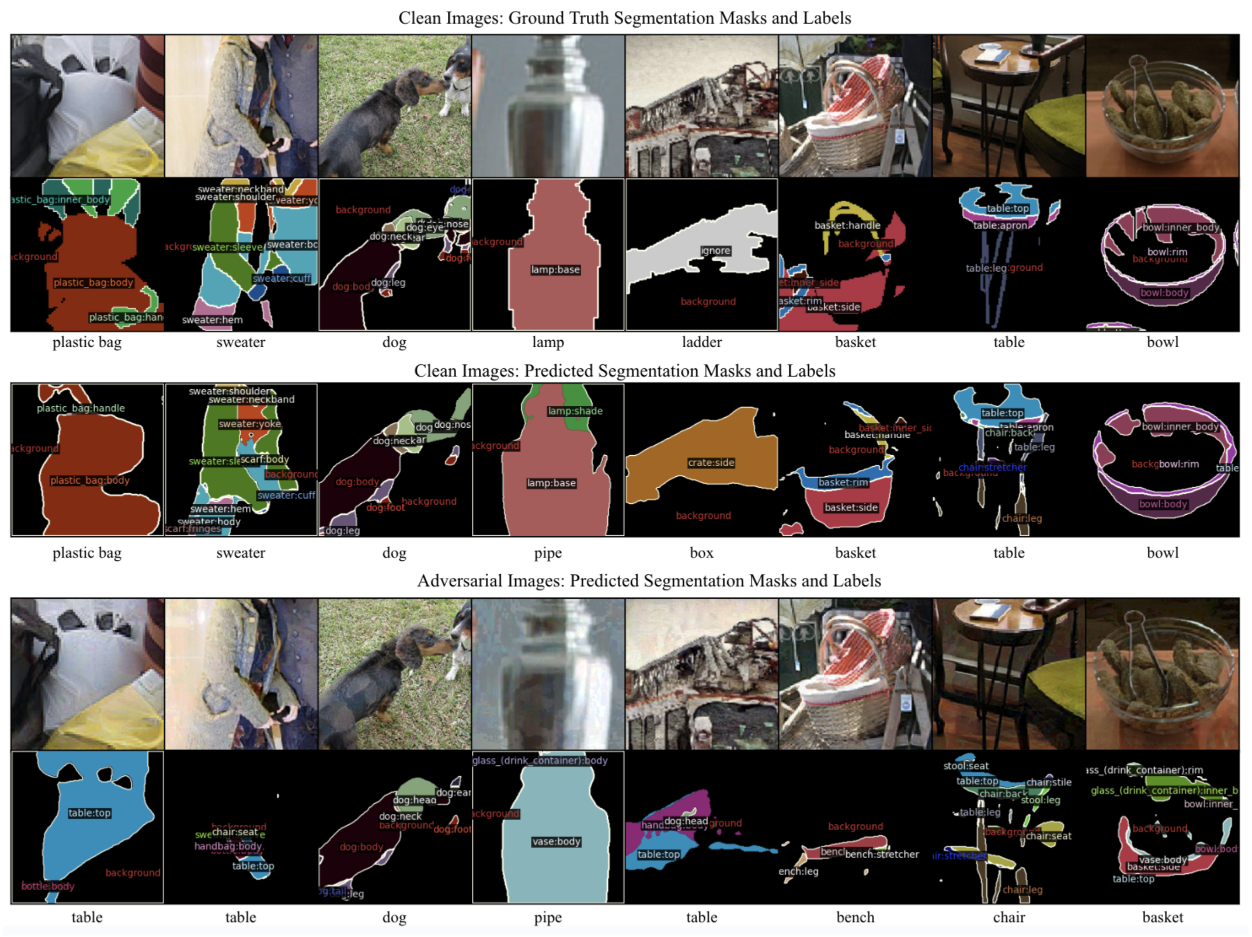
Clean Images: Ground Truth Segmentation Masks and Labels
plastic bag
sweater
dog
lamp
ladder
basket
table
bowl
Clean Images: Predicted Segmentation Masks and Labels
plastic bag
sweater
dog
pipe
box
basket
table
bowl
Adversarial Images: Predicted Segmentation Masks and Labels
table
table
dog
pipe
table
bench
chair
basket

Figure 3.4: Predictions for Part-based Model with Mask DINO

# Chapter 4

# Conclusion

In this study, we have demonstrated that it is possible to achieve adversarial robustness through the use of part bounding box labels instead of expensive fine-grained segmentation masks. By providing a more cost-effective form of supervision to models, our approach has the potential to significantly reduce annotation costs and improve the practical applicability of part-based models in real-world scenarios.

Our experimental results indicate that DeepLabv3+ based models perform better than DINO-based models for achieving robust classification in an adversarial setting. Counter-intuitively, while DINO-based models exhibit good performance on clean samples, their performance significantly deteriorates on adversarial samples. This phenomenon is seen consistently though out our experiments with both the DINO object detector and the Mask DINO segmenter. We encourage future work to explore strategies for improving the adversarial training of transformer-based models.

Finally, we have shown that part-based models can be scaled to a larger and more diverse dataset like PACO. In particular, our part-based models show significant improvements in both clean and robust accuracy as compared to baseline models that do not use additional supervision signals.

# Bibliography

[1] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV].

[2] Liang-Chieh Chen et al. *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. 2018. arXiv: 1802.02611 [cs.CV].

[3] Sven Gowal et al. *Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples*. 2021. arXiv: 2010.03593 [stat.ML].

[4] Ju He et al. *PartImageNet: A Large, High-Quality Dataset of Parts*. 2022. arXiv: 2112.00933 [cs.CV].

[5] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[6] Feng Li et al. *Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation*. 2022. arXiv: 2206.02777 [cs.CV].

[7] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[8] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV].

[9] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML].

[10] Vignesh Ramanathan et al. “PACO: Parts and Attributes of Common Objects”. In: *arXiv preprint arXiv:2301.01795*. 2023.

[11] Chawin Sitawarin et al. *Part-Based Models Improve Adversarial Robustness*. 2023. arXiv: 2209.09117 [cs.CV].

[12] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

[13] Hao Zhang et al. *DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection*. 2022. arXiv: 2203.03605 [cs.CV].