Perception for Real-World Robotic Applications

YuXuan Liu
Pieter Abbeel

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-122
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-122.html

May 12, 2023

Copyright © 2023, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Perception for Real-World Robotics Applications

by

Yu Xuan Liu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel, Chair
Professor Ken Goldberg
Professor David Whitney

Spring 2023

Perception for Real-World Robotics Applications

Copyright 2023
by
Yu Xuan Liu

Abstract

Perception for Real-World Robotics Applications

by

Yu Xuan Liu

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

Recent advances in artificial intelligence, particularly deep learning and large foundation models, have demonstrated remarkable progress. However, when applying AI to real-world robotics applications, we still face many challenges due to the diverse scenarios and objects encountered, as well as the need for high throughput and accuracy. Off-the-shelf models often fail to meet the stringent requirements for high-performing robotic applications, because they do not adequately model uncertainty that arises in the real world. Moreover, training such models require large datasets which can be expensive to annotate or not immediately applicable for robotic applications.

We address these challenges by introducing a novel class of models that explicitly model and handle ambiguity in 2D and 3D perception. These models offer improved adaptability and decision-making capabilities by incorporating uncertainty estimation, better equipping robots for the dynamic nature of real-world environments. Furthermore, we explore methods of leveraging diverse data collected in robotic applications without requiring costly human annotation. We propose a self-supervised learning method that enables robots to autonomously learn from the rich information available in the diverse images they encounter during operation. This approach leads to enhanced performance and adaptability, allowing robotic systems to continuously refine their perception capabilities. We hope these contributions pave the way for more robust, adaptable, and high-performing robotic systems that excel in complex and dynamic environments, addressing the unique challenges posed by real-world robotics and bridging the gap between AI research and practical robotic applications.

To all my friends, family, colleagues, teachers, and the Covariant team.

Contents

Chapter 1

Introduction

1.1 Real-World Robotics

Industrial robotics has been a cornerstone of modern manufacturing and automation for many decades. Traditionally, these robots perform pre-programmed tasks, acting blindly and repetitively, executing the same operations over and over again. While this approach works well for specific tasks in controlled environments, it falls short when attempting more complex tasks, such as picking objects out of a bin, which require greater adaptability and perception capabilities due to the diversity of items and numerous edge cases encountered.

Recent advancements in robotics research have focused on enabling robots to solve general tasks using natural language, exemplified by systems like RT-1[6]. However, these models often lack the speed and reliability necessary for real-world industrial applications. Warehouse and industrial settings demand exceedingly high accuracy, often exceeding 99%, and human-level throughput to meet the efficiency requirements of modern production lines.

In this thesis, we investigate existing models in the literature and identify their shortcomings when applied to real-world industrial use-cases. We delve into the challenges faced by off-the-shelf models, which often struggle with uncertainty and ambiguity, resulting in suboptimal performance in diverse and dynamic environments.

To address these limitations, we propose new models that push the boundaries of current state-of-the-art perception and decision-making capabilities in robotic systems. Our research focuses on developing models that can handle ambiguity and uncertainty, allowing for improved adaptability and performance in real-world scenarios. Moreover, we explore self-supervised learning methods to leverage diverse data collected in real robotic applications without the need for costly human annotation, enabling robots to learn and refine their perception capabilities autonomously.

The contributions of this thesis aim to bridge the gap between AI research and practical robotic applications in industrial settings. By proposing novel models and methodologies tailored for real-world robotics, we strive to advance the development of more robust, adaptable, and high-performing robotic systems that can tackle the complex challenges inherent

in diverse and dynamic environments.

1.2 Overview

In this thesis, we address the challenges of real-world robotic applications by proposing new models that can effectively model uncertainty and leverage self-supervised data for learning. Specifically, we introduce three novel methods: (1) Autoregressive bounding box prediction, (2) Latent-MaskRCNN for instance segmentation, and (3) Self-supervised instance segmentation. These methods build upon state-of-the-art techniques to better capture and handle ambiguity, providing a more robust and adaptable foundation for robotic systems operating in dynamic and complex environments.

1. Autoregressive Bounding Box Prediction

In the first part of this thesis, we tackle the problem of 3D bounding box prediction, a widespread intermediate representation in many computer vision applications. Predicting 3D bounding boxes is challenging due to partial observability and the need for a strong sense of uncertainty. We propose an autoregressive prediction head that enhances the modeling of the output distribution, allowing the model to capture meaningful uncertainty measures. Our method achieves strong results on various datasets, including SUN-RGBD, Scannet, KITTI, and our newly released COB-3D dataset, which highlights new types of ambiguity arising in real-world robotics applications. This section represents work published in Yuxuan Liu et al. "Autoregressive Uncertainty Modeling for 3D Bounding Box Prediction". In: *Computer Vision - ECCV 2022 - 17th European Conference*. 2022.

2. Latent-MaskRCNN for Instance Segmentation

In the second part of the thesis, we focus on instance segmentation, a fundamental skill for many robotic applications. We propose Latent-MaskRCNN, a class of distributional instance segmentation models using latent codes that can model uncertainty over plausible hypotheses of object masks. By incorporating uncertainty estimation into instance segmentation, our model significantly reduces critical errors in robotic systems, including those encountered in our newly released dataset of ambiguous scenes in a robotic application. In real-world apparel-picking robots, our method demonstrates a significant reduction in double pick errors while maintaining high performance. This section represents work published in Yuxuan Liu et al. "Distributional Instance Segmentation: Modeling Uncertainty and High Confidence Predictions with Latent-MaskRCNN". in: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023.

3. Self-Supervised Instance Segmentation from Grasping

In the third and final part of the thesis, we explore the self-supervised learning methods for instance segmentation in robotic applications. We propose a method that leverages grasp interactions to collect segmentation supervision, enabling robots to learn from thousands of grasp interactions without costly human annotation. By segmenting grasped objects and using a "cut-and-paste" generation method, instance segmentation models trained with our method achieve better performance than those trained with 10x the amount of labeled data. On a real robotic grasping system, our instance segmentation model reduces the rate of grasp errors by over 3x compared to an image subtraction baseline. This section represents work published in YuXuan Liu, Xi Chen, and Pieter Abbeel. *Self-Supervised Instance Segmentation by Grasping*. 2023. arXiv: `2305.06305 [cs.CV]`.

Chapter 2

Autoregressive 3D Bounding Box

2.1 Introduction

Predicting 3D bounding boxes is a core part of the computer vision stack in many real world applications, including autonomous driving, robotics, and augmented reality. The inputs to a 3D bounding box predictor usually consist of an RGB image and a point cloud; the latter is typically obtained from a 3D sensor such as LIDAR or stereo depth cameras. These 3D sensing modalities have their own idiosyncrasies: LIDAR tends to be accurate but very sparse, and stereo depth can be both sparse and noisy. When combined with the fact that objects are only seen from one perspective, the bounding-box prediction problem is fundamentally underspecified: the available information is not sufficient to unambiguously perform the task.

Imagine that a robot is going to grasp an object and manipulate it — understanding the uncertainty over the size can have a profound impact on what the robot decides to do next. For example, if it uses the predicted bounding box to avoid collisions during motion planning, then we may want to be conservative and err on the larger side. However, if it is trying to pack the items into a shipment, then having accurate dimensions may also be important.

Consider the scene depicted in Figure 2.1, which we observed in a real-world robotics application. From the image of the object in a), it is fairly easy to gauge the width and length of the indicated object, but how tall is it? The object could be as deep as the bin, or it could be a stack of two identical objects, or even a thin object – but from the available information, it is impossible to say for sure. Formulating bounding box prediction as a regression problem results in a model that can only make a "pointwise" prediction – even in the face of ambiguity, we will only get a single predicted bounding box, shown in b).

A sufficiently expressive bounding-box model should be able to output the entire range of plausible bounding box hypotheses and make different predictions for different confidence requirements. A 0.5-confidence box d) must contain the object 50% of the time while a 0.8-confidence box e) will expand in the direction of uncertainty to contain the object 80%

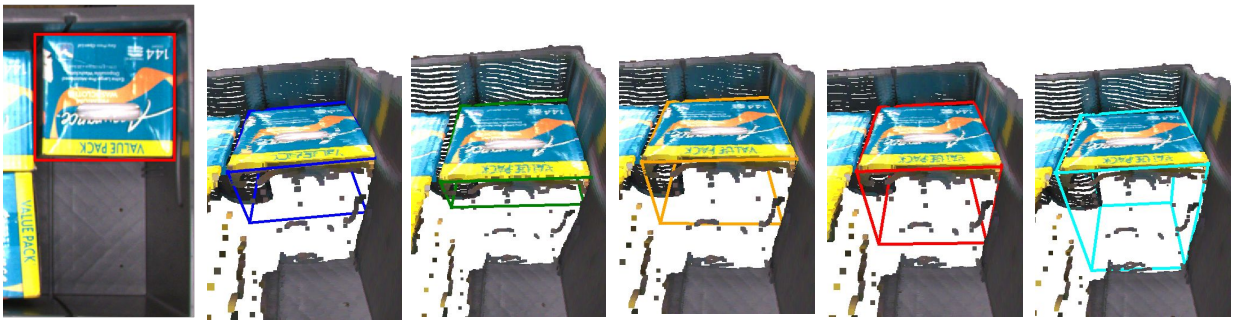

Figure 2.1: a) In this scene from a real-world robotics application, how tall is the object highlighted in red? b) A pointwise model could output only one box prediction with no notion of uncertainty c)-e) Predictions from our confidence box method. Notice that the predicted box expands in the direction of uncertainty as we increase the confidence requirement. f) Our dimension conditioning method can leverage additional information to make more accurate predictions.

of the time. Moreover, such a model could leverage additional information, such as known dimensions of an object, to make even more accurate predictions, as shown in f).

Setting aside partial observability, the prediction space has complexities that require care in the design of a bounding-box estimator. Making accurate predictions requires the estimator to reason about rotations, which has been observed to be notoriously difficult for neural networks to predict and model uncertainty over [98, 23, 69]. Many existing methods sidestep this problem by constraining their predictions to allow rotation about a single axis or no rotations at all. This can be sufficient for some applications but has shortcomings for the general case.

A common thread that links these challenges together is the necessity to reason about uncertainty. This has been largely underexplored in existing work, but we hypothesize that it is critical to improving 3D bounding box estimators and expanding their usability in applications of interest. We propose to tackle this problem by predicting a more expressive probability distribution that explicitly accounts for the relationships between different box parameters. Using a technique that has proven effective in other domains, we propose to model 3D bounding boxes autoregressively: that is, to predict each box component sequentially, conditioned on the previous ones. This allows us to model multimodal uncertainty due to incomplete information, make high confidence predictions in the face of uncertainty, and seamlessly relax the orientation constraints that are popular in existing methods. To summarize our contributions:

1. We propose an autoregressive formulation to 3D bounding box prediction that can model complex, multimodal uncertainty. We show how this formulation can gracefully

scale to predict complete 3D orientations, rather than the 0- or 1-D alternatives that are common in prior work.

2. We propose a method to make high confidence predictions in ambiguous scenarios and estimate useful measures of uncertainty.

3. We introduce a simulated dataset of robotics scenes that illustrates why capturing uncertainty is important for 3D bounding box prediction, as well as the benefits and challenges of predicting full 3D rotations.

4. We show that our formulation applies to both traditional 3D bounding box estimation and 3D object detection, achieving competitive results on popular indoor and autonomous driving datasets in addition to our dataset.

2.2 Related Work

3D Bounding-box Estimation: Early work on 3D bounding box prediction [64, 70] assumes that object detection or segmentation has already been performed, and the bounding box predictor solely needs to identify a single 3D bounding box within a filtered point cloud. We refer to this task as *3D bounding-box estimation*. Much of this work focused on developing architectures to easily consume point cloud data, which often can be sparse and/or unstructured when obtained from real-world data.

3D Object Detection: Recently, a number of methods [79, 76, 54, 63, 90, 80, 71] have explored how to jointly perform object detection and 3D bounding box estimation, rather than treating them as two explicit steps. This task is known as *3D object detection* and is quickly gaining popularity over the decoupled detection and estimation tasks. The main focus is on how to take the network architectures that have proven successful at the estimation task (which have strong inductive biases for operating on point clouds), and combine them with the architectures commonly used for the 2D object detection method (which are usually based on region proposals).

Uncertainty Modeling in Object Detection: Uncertainty modeling has been studied in the context of 2D and 3D Object Detection [60, 46, 97, 59, 28]. In many cases, these methods will use independent distributions, such as Gaussian or Laplace, to model uncertainty over box parameters such as corners, dimensions, and centers [33, 60, 13]. While these distributions may capture some uncertainty for simple box parameterizations, they don't capture correlations across parameters and have yet to be proven on full 3D rotations.

Autoregressive Models: Deep autoregressive models are frequently employed across a variety of domains. In deep learning, they first gained popularity for generative modeling of images [86, 66, 87], since they can model long-range dependencies to ensure that pixels later in the autoregressive ordering are sampled consistently with the ones sampled earlier. In addition to being applied to other high-dimensional data such as audio [66], they have

also been shown to offer precise predictions even for much lower-dimensional data, such as robot joint angles or motor torques [58].

2.3 Autoregressive 3D Bounding Box Prediction

3D bounding box estimation is typically formulated as a regression problem over the dimensions $d = (d_x, d_y, d_z)$, center $c = (c_x, c_y, c_z)$, and rotation $R = (\psi, \theta, \phi)$ of a bounding box, given some perceptual features h computed from the scene, e.g. from an image and point cloud. Prior work has explored various parametrizations and loss functions, but a notable salient feature to observe is that they all predict a *pointwise* estimate of the bounding box: the model simply outputs all of the box parameters at once. In 3D object detection, such regression is typically applied to every box within a set of candidates (or *anchors*), and fits into a larger cascade that includes classifying which anchors are relevant and filtering out unnecessary or duplicate anchors. In practice, this formulation can be greatly limiting, especially in the face of partial observability or symmetry.

Autoregressive Modeling

We propose to tackle this problem by autoregressively modeling the components of a 3D bounding box. That is, for some ordering of the components (e.g. dimensions $\rightarrow$ center $\rightarrow$ orientation, or any permutation thereof), such a predictor will sequentially predict each component conditioned on the previous ones. In theory, the particular autoregressive ordering should not matter; empirically, we find that dimensions $\rightarrow$ center $\rightarrow$ orientation was effective, so we use this ordering for our model. Having dimension as first in the autoregressive ordering also enables us to condition on dimensions when they are known which can be effective at improving the prediction accuracy.

We discretize the box parameters rather than predicting continuous values, which is a well-known technique that allows the model to easily express multimodal distributions [86]. For rotations, we chose Euler angles since each dimension has a fixed range and does not to be normalized. To make discrete dimension and center predictions, we normalize those parameters so that they can fit within a fixed set of discrete bins. We normalize dimensions by some scale s so that most values of d/s are within the range $[0, 1]$, and offset the centers by c_0 so that most normalized centers $(c - c_0)/s$ are within the range $[-1, 1]$. We use 512 bins for each dimension and adjust the bin range to achieve on average ≥ 0.99 IOU with the quantized box and $< 0.1\%$ overflow or underflow due to quantization.

From RGB-D inputs we extract a fixed-dimensional feature vector h for each object. For each parameter $b = (d_x, d_y, d_z, c_x, c_y, c_z, \psi, \theta, \phi)$ in the autoregressive ordering, we model $p(b_i|b_1, \ldots, b_{i-1}, h)$ using a MLP with 2-3 hidden layers. This autoregressive model is then

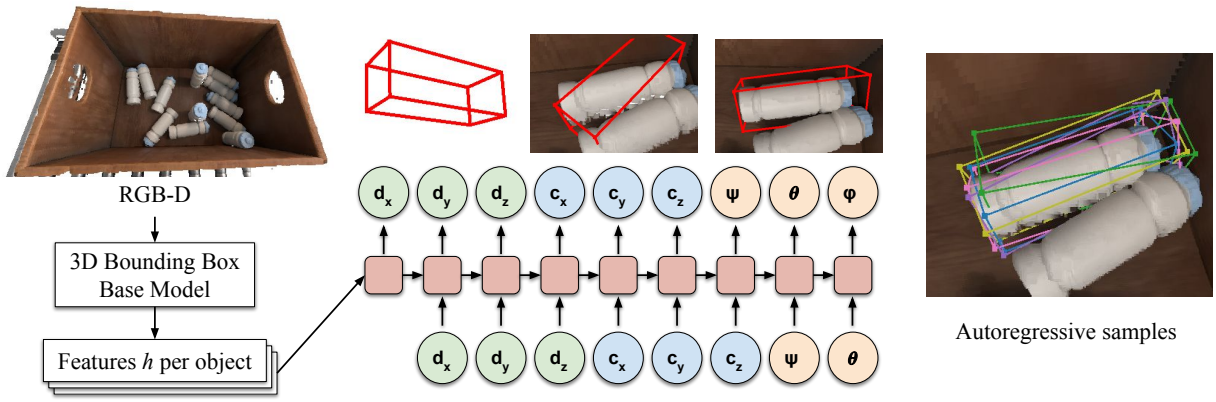

Figure 2.2: We compute per-object features h using a base model from RGB-D input. Then, we autoregressively sample dimensions, center, and rotations, each step conditioned on the previous one. We can express uncertainty through samples, such as the rotational symmetry of the bottle, whereas pointwise models could only make a single prediction.

trained using maximum likelihood:

$$\log p(b|h) = \sum_{i=1}^{9} \log p(b_i|b_1, \ldots, b_{i-1}, h) \tag{2.1}$$

Model Architectures

Our autoregressive prediction scheme can be applied to any type of 3D bounding box predictor. In this section, we discuss how it might be applied in two different contexts: 3D object detection and 3D bounding box estimation.

Autoregressive 3D Object Detection.

FCAF3D [76] is a state-of-the-art 3D object detection method that was heavily engineered to exploit sparse and unstructured point clouds. Given a colored point cloud, it applies a specialized feature extractor consisting of sparse 3D convolutions, and then proposes 3D bounding boxes following a popular single-stage detector, FCOS [84].

Autoregressive FCAF3D: We can make FCAF3D autoregressive by adding a head and training this head with maximum likelihood in addition to the FCAF3D loss $L_F(h, y)$ (Figure 2.3). We found that the pointwise box prediction was useful to condition the autoregressive prediction and estimate the scaling normalization factor $s = \max\{d'_x, d'_y, d'_z\}$, where d' is the pointwise dimension prediction of FCAF3D. Bounding box centers c are normalized by the output locations c_0 of the sparse convolutions and scaled by the same s: $(c - c_0)/s$.

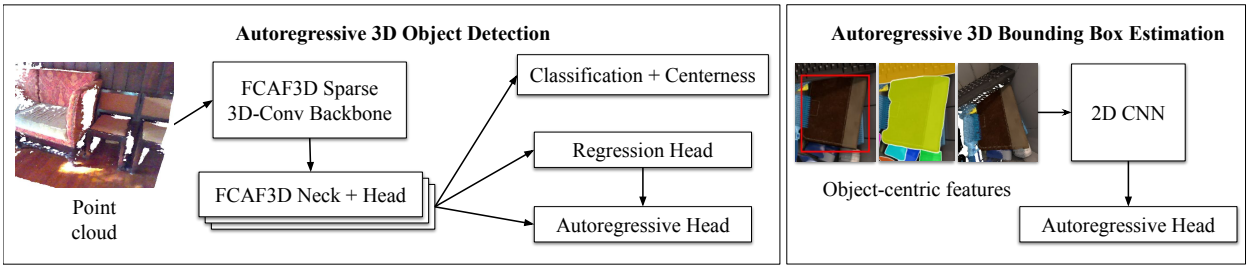

Figure 2.3: For indoor 3D Object Detection, we use FCAF3D as a base model with an autoregressive head for bounding box prediction. For 3D Bounding Box Estimation we take object-centric features from a 2D object detector and pass them into a 2D CNN for autoregressive bounding box prediction.

Since 3D object detection datasets have at most one degree of freedom for rotation, we predict only one θ parameter for box rotation.

To optimize the autoregressive prediction for higher IOU, we sample boxes $b \sim p(b|h)$ and maximize the IOUs of the samples with the ground truth box y. For this optimization, we use the conditional expectation b' where $b'_i = \mathbb{E}[b_i|b_1, \ldots, b_{i-1}, h]$ (since b' is differentiable) to maximize $IOU(b', y)$. Altogether, we train autoregressive FCAF3D using the combined loss:

$$L(h, y) = L_F(h, y) - \log p(b|h) + \mathbb{E}_{b\sim p(b|h)}[1 - IOU(b', y)] \tag{2.2}$$

Autoregressive PV-RCNN: Lidar-based object detection networks, such as PV-RCNN [80], typically have different architectures and inductive biases than indoor detection models. However, we show that our autoregressive box parameterization is agnostic to the underlying architecture by applying it to PV-RCNN. We propose Autoregressive PV-RCNN by extending the proposal refinement head to be autoregressive, modeling the residual Δr^α as discrete autoregressive $p(\Delta r^\alpha|h)$. Then, we add $-\log p(\Delta r^\alpha|h)$ to the total training loss.

Autoregressive 3D Bounding Box Estimation.

3D Bounding Box Estimation assumes that object detection has already been performed in 2D, and we simply need to predict a 3D bounding box for each detected object. To highlight that our autoregressive prediction scheme can be applied to any bounding box predictor, we chose a model architecture that is substantially different from FCAF3D. For each detected object, we take an object-centric crop of the point cloud, normals, and object mask as input to a 2D-CNN, producing a fixed feature vector h per object. This h is used as features for our autoregressive parameterization $p(b|h)$. See Appendix A.1 for more details on the architecture.

To normalize the input and box parameters, we scale by the range of the first and third quartiles of each point cloud dimension $s = Q_3 - Q_1$, and recenter by the mean of the

quartiles $c_0 = \frac{Q_1+Q_3}{2}$. For full SO(3) rotations, we found there were many box parameters that could represent the same box; for example, a box with $d = (1, 2, 3)$ is equivalent to a box with $d' = (2, 1, 3)$ and a 90° rotation. To account for this, we find all the box parameters $B = \{b^{(1)}, ..., b^{(m)}\}$ that represent the same box and supervise on all of them:

$$L(h, B) = -\frac{1}{|B|} \sum_{b^{(i)} \in B} \log(b^{(i)}|h) \tag{2.3}$$

2.4 Applying Autoregressive 3D Bounding Box Models

Given a trained autoregressive bounding-box model, how do we actually obtain predictions from it? There can be a few different options, depending on how the downstream application plans to use the predictions.

Beam Search

In many applications, we want to simply obtain the most likely 3D bounding box given the input observation. That is, we find the box $b^* = \arg\max_b p(b|h)$ which is most likely under the model. Finding b^* exactly can be computationally expensive, but we can approximate it using *beam search*, a technique that has proven especially popular for autoregressive models in natural language applications [20]. Beam search allows us to estimate the mode of the distribution learned by the model and serves as an effective pointwise prediction.

Quantile and Confidence Boxes

In applications such as robotics and autonomous driving, 3D bounding boxes are often used to estimate object extents and avoid collisions. To that end, we often care that an object o is fully contained in the estimated box b. For a given confidence requirement p, we define a confidence box b_p as a box that contains the true object o with probability at least p: $\mathbb{P}(o \subseteq b_p) \geq p$. We'll show how to use an autoregressive bounding box model for confidence box predictions.

Suppose we draw multiple samples K from our model. If a point $x \in \mathbb{R}^3$ is contained in many boxes, then it's likely that point is actually part of the object. Conversely, a point that is only contained in a few sampled boxes is not likely to be part of the object. We can formalize this intuition as the occupancy measure

$$O(x) = \mathbb{P}(x \in b) = \mathbb{E}_{b \sim p(b|h)}[\mathbb{1}\{x \in b\}] \approx \frac{1}{K} \sum_{i=1}^{K} \mathbb{1}\{x \in b^{(i)}\} \tag{2.4}$$

which can be approximated using samples $b^{(1)}, \ldots, b^{(K)} \sim p(b|h)$ from our model.

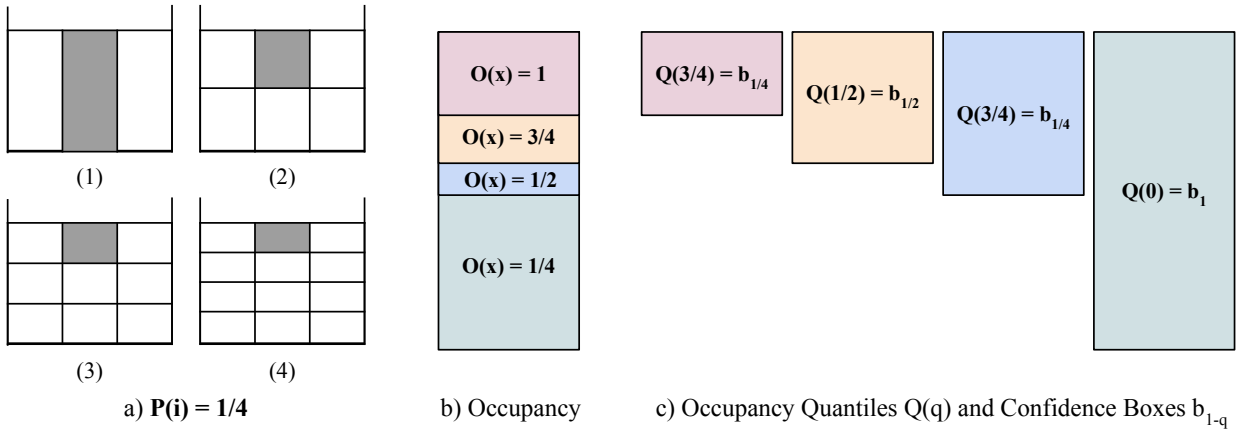

Figure 2.4: Consider a scenario where we are estimating the bounding box of a tightly packed bin of stacked boxes. a) There is not enough visual information to estimate the object height, however, we know that the object could have heights H/i for $i \in \{1, 2, 3, 4\}$ with equal probability. b) We compute the occupancy $O(x)$ for different regions. c) We visualize occupancy quantiles $Q(q)$ which correspond to confidence boxes b_{1-q}. Notice that as the confidence requirement increases, the size of the box increases to ensure we can contain the true object.

To find regions that are very likely to be part of an object, consider the set of all points that have occupancy greater than q:

$$Q(q) = \{x : O(x) > q\} \tag{2.5}$$

which we'll refer to as the *occupancy quantile*. The minimum volume bounding box over the occupancy quantile is the *quantile box*:

$$b_q = \arg \min_{b:Q(q)\subseteq b} \text{vol}(b) \tag{2.6}$$

Under some conditions, we can show that quantile boxes are confidence boxes.

Theorem 1 *A quantile box with quantile q is a confidence box with confidence $p = 1 - q$ when $p(b|h)$ is an ordered object distribution.*

$p(b|h)$ is an ordered object distribution if for any two distinct boxes b_i, b_j in the sample space of $p(b|h)$, one box must be contained within the other, $b_i \subset b_j$ or $b_j \subset b_i$. Empirically we find that quantile boxes are good approximations for confidence boxes even when $p(b|h)$ is not an ordered object distribution. See Figure 2.4 for a visualization of occupancy and confidence boxes.

Quantile boxes provide an efficient way to make confidence box predictions with an autoregressive model. We can use the autoregressive distribution to estimate occupancy using

supervision from 3D box labels (without requiring meshes for direct occupancy supervision). Occupancy quantiles provide a fast approach for confidence box estimation on ordered object distributions and a good confidence box approximation for general object distributions. Appendix A.2 has the full proof of Theorem 1 and the details of our fast quantile box algorithm.

Uncertainty Measure

Uncertainty estimation is an important application of bounding box estimation. When the 3D extent of an object is unknown or not fully observed, it can be valuable if a model can also indicate that its predictions are uncertain. For instance, a robot may choose to manipulate that uncertain object more slowly to avoid collisions, or an autonomous vehicle may be more cautious around a moving object of unknown size.

A pointwise predictor can accomplish this by predicting both a mean μ and variance σ^2 for each box parameter, maximizing a $\mathcal{N}(\mu, \sigma^2)$ likelihood [33]. However, the spread of the distribution is measured independently for each box parameter which doesn't measure the spread of the overall box distribution well.

With an autoregressive box parameterization, we can measure uncertainty in the space of boxes using quantile boxes. Let b_α and b_β be two quantile boxes with different quantiles. If we consider these boxes as confidence boxes, we can interpret (b_α, b_β) as a confidence interval or the spread of the box distribution. With this intuition, we can measure uncertainty using the IOU of different quantile boxes $U_{\alpha,\beta} = 1 - IOU(b_\alpha, b_\beta)$. This $U_{\alpha,\beta}$ effectively measures the span of the distribution in units of relative volume.

Dimension Conditioning

For some robotics applications, such as object manipulation in industrial settings, we are often presented with Stock-Keeping Unit, or SKU, information beforehand. In these scenarios, the dimensions of each SKU are provided, and the prediction task essentially boils down to correctly assigning the dimensions to a detected object instance, and predicting the pose of the 3D bounding box.

The autoregressive nature of our model allows for conveniently conditioning on the dimensions of each bounding box. However, we don't know which object in the scene corresponds to which SKU dimensions. How can we leverage dimension information from multiple SKUs without object-SKU correspondence? Our autoregressive model provides an elegant solution using conditioning and likelihood evaluation.

Given $\{d^{(1)}, ..., d^{(k)}\}$ known SKU dimensions, we can make a bounding box prediction using this information by maximizing:

$$b^* = \arg\max_b \{ \max_{d^{(1)},...,d^{(k)}} p(b|d^{(i)}, h) \} \tag{2.7}$$

We can find the optimal b^* by using beam search conditioned on each of the d_i and returning the box with the highest likelihood. Figure 2.1 shows an illustrative example of how dimension conditioning can be used to greatly increase the fidelity of the predicted 3D bounding boxes.

2.5 Experiments

We designed our experiments to answer the following questions:

1. How does an autoregressive bounding box predictor perform compared to a pointwise predictor, across a variety of domains and model architectures?

2. How meaningful are the uncertainty estimates from an autoregressive model? Are quantile boxes confidence boxes for general object distributions?

Datasets

To demonstrate the flexibility of our method, we conducted experiments on a diverse set of indoor, outdoor, and industrial datasets:

SUN-RGBD [82] is a real-world dataset containing monocular images and point clouds captured from a stereo depth camera. It features a large variety of indoor scenes and is one of the most popular benchmarks in 3D object detection. The box labels only include one rotational degree of freedom θ.

Scannet [15] is a dataset of indoor 3D reconstructions. There are 18 classes and box labels are axis-aligned (no rotation). We train on 1201 scenes and evaluate on 312 validation scenes.

KITTI [22] is a widely popular 3D detection dataset for autonomous driving. Objects in KITTI have one degree of rotational freedom θ, and we report evaluation results on the validation split.

COB-3D. *Common Objects in Bins 3D* is a simulated dataset rendered by Theory Studios to explore a qualitatively different set of challenges than the ones exhibited in popular datasets in the literature. We are releasing nearly 7000 scenes that aim to emulate industrial order-picking environments with each scene consisting of a bin containing a variety of items. There are two main themes we chose to highlight: first, the objects are in a greater range of orientations than any other 3D-bounding-box dataset. In particular, a model that performs well must reason about complete 3D rotations, whereas the state-of-the-art methods on SUN-RGBD only need to predict one rotational degree of freedom. Secondly, it exhibits many types of ambiguity including rotation symmetry, occlusion reasoning in cluttered scenes, and tightly-pack bins with unobserved dimensions. See Appendix A.3 for full details on this dataset including visual examples.

Evaluation

To evaluate 3D-bounding-box predictions, *intersection-over-union*, or IoU, is commonly used to compare the similarity between two boxes. 3D object detection uses mean average precision, or mAP, to measure how well a detector trades off precision and recall. IoU is used to determine whether a prediction is close enough to a ground-truth box to constitute a true positive. For 3D bounding-box estimation, detection has already happened, so we simply measure the mean IoU between the prediction and ground-truth, averaged across objects.

Unlike 2D detection, many applications that use 3D bounding boxes especially care about underestimation more than overestimation: if the predicted bounding box is too large, that is generally a less costly error than if it is too small. In the latter case, there are parts of the object that are outside the bounding box, which may result in collisions in robotics or autonomous driving setting.

To help quantify this error asymmetry, we consider a new similarity functions, the *intersection-over-ground-truth* (IoG). IoG measure what fraction of the ground truth box is contained within the predicted box; when IoG is 1, the ground truth box is fully contained in the predicted box. With IoG and IoU, we have a more complete understanding of the types of errors that a bounding-box predictor is making. For the detection task, we compute mAP separately using IoU and IoG, and for the estimation task, we compute the mean IoG along with the mean IoU.

3D Object Detection

To evaluate the autoregressive box parameterization for 3D Object Detection, we evaluate Autoregressive FCAF3D and Autoregressive PV-RCNN introduced in Section 2.3. Table 2.1 shows the comparison between autoregressive models and baselines on SUN-RGBD, Scannet, and KITTI. We find that beam search generally matches the baseline performance, if not exceeding performance on IoU AP_{all}.

As for quantile boxes, we find that lower quantiles result in higher IoG mAP which suggests that the predicted boxes are more likely to contain the ground truth box. This is consistent with our claim from Theorem 1 since lower quantiles correspond to higher confidence boxes and must contain the true object with higher probability. We find that quantile boxes 0.4-0.5 strike the best balance between IoU and IoG, achieving better mAP than baselines in most cases. This flexible quantile parameter enables applications to trade off bounding box accuracy as measured by IoU with containment probability as measured by IoG. For instance, an autonomous vehicle may use a lower quantile to mitigate the risk of collisions at the cost of some bounding box accuracy.

3D Bounding Box Estimation

We evaluate the bounding box estimation on COB-3D using the model architecture described in Section 2.3. To compare the effectiveness of our autoregressive parameterization, we train

Table 2.1: 3D Object Detection results on SUN-RGBD, Scannet, and KITTI

Dataset	Method	IoU $AP_{0.25}$	IoU $AP_{0.50}$	IoU AP_{all}	IoG $AP_{0.25}$	IoG $AP_{0.50}$	IoG AP_{all}
SUN-RGBD	FCAF3D	63.8	48.2	37.42	64.72	59.82	48.75
	3DETR	59.52	32.17	31.13	63.00	53.33	44.08
	VoteNet	60.71	38.98	30.25	62.81	54.58	43.62
	ImVoteNet	**64.24**	39.38	31.12	**67.00**	57.41	45.78
	Beam Search	62.94	47.03	38.15	64.75	58.50	47.17
	Quantile 0.1	61.21	30.94	31.06	65.89	**64.34**	**60.08**
	Quantile 0.4	63.46	48.41	38.43	65.34	61.68	51.76
	Quantile 0.45	63.47	**48.64**	**38.55**	65.19	61.03	50.36
	Quantile 0.5	63.30	47.70	38.50	64.99	59.83	48.44
Scannet	FCAF3D	68.53	**53.87**	43.32	72.05	67.63	60.66
	3DETR	64.09	47.16	39.57	68.62	59.17	49.82
	Beam Search	**69.06**	53.67	**43.85**	71.46	66.10	59.13
	Quantile 0.1	67.10	43.13	34.17	72.23	**70.01**	**66.73**
	Quantile 0.2	68.03	48.68	38.27	**72.30**	69.68	65.43
	Quantile 0.4	68.73	52.98	42.76	72.08	67.74	61.98
KITTI	Method	AP IoU Hard Split Car	AP IoU Hard Split Ped.	AP IoU Hard Split Cycl.	AP IoG Hard Split Car	AP IoG Hard Split Ped.	AP IoG Hard Split Cycl.
	PVRCNN	**82.37**	53.12	68.69	91.86	67.08	73.14
	Beam Search	**82.37**	52.28	**69.13**	91.84	66.96	73.40
	Quantile 0.1	59.75	39.26	58.38	**96.02**	**71.85**	**76.09**
	Quantile 0.4	81.98	**54.15**	68.45	93.98	70.63	74.08
	Quantile 0.5	82.32	53.78	69.03	91.84	68.14	73.52

the same model architecture with different box parameterizations and losses. All models receive the same 2D detection results and features as input and must make 3D bounding box predictions for each detected object. We consider 4 baseline parameterizations for this task inspired by various works in the literature:

L1 Regression: In this parameterization, the model outputs 9 real values for each of the 9 box parameters: $b = (d_x, d_y, d_z, c_x, c_y, c_z, \psi, \theta, \phi)$. The model predicts dimensions and centers in coordinates normalized around the object's point cloud. This model is trained using a L1 loss over the normalized box parameters $L(b, g) = ||b - g||_1$, where g is the ground truth box [63].

Gaussian: For this baseline, the model outputs 18 real values for the mean, μ, and log-variance, $\log \sigma^2$, of 9 Gaussian distributions $\mathcal{N}(\mu, \sigma)$ over the box parameters b [33, 60]. Predicting the variance enables the model to output uncertainty over different box parameters, independently of each other. We train this model using maximum likelihood:

Table 2.2: Results of the proposed method & baselines on our dataset. We also show results for conditioning our method on ground truth dimensions

	IoU	IoG	F1	$err_{dim}[m]$	$err_{quat}[rad]$	$err_{center}[m]$
L1 Regression	0.4219	0.6113	0.4992	0.0436	0.4667	0.0138
Discrete	0.5232	0.6282	0.5709	0.0339	0.2926	**0.0105**
Gaussian	0.3169	0.5304	0.3967	0.0450	0.5154	0.0119
4-Point	0.5688	0.7113	0.6321	0.0332	0.1999	0.0132
Beam Search	**0.6296**	0.7877	0.6999	**0.0287**	**0.1598**	0.0109
Quantile 0.1	0.3821	**0.9723**	0.5486	0.0986	0.1762	0.0123
Quantile 0.4	0.5949	0.8871	0.7122	0.0377	0.1640	0.0110
Quantile 0.5	0.6275	0.8295	**0.7126**	0.0318	0.1657	0.0110
Conditioning	0.6709	0.7899	0.7215	0.0086	0.1674	0.0096

$L(\mu, \log \sigma^2, g) = -\sum_i \log \mathcal{N}(g_i; \mu_i, \sigma_i)$.

Discrete: In some prior works, box parameters are predicted as discrete bins but not in an autoregressive manner [72]. To evaluate this parameterization and ablate the necessity of autoregressive predictions, we predict each box parameter *independently* as discrete bins: $\log p(b|h) = \sum_{i=1}^{9} \log p(b_i|h)$

4-Point: This baseline outputs 12 real values for four 3D corner points $(p_0, p_1, p_2, p_3) \in \mathbb{R}^3$, constituting a 3D bounding box [60, 59]. We ensure that the 3D bounding box is orthogonal by applying the Gram-Schmidt process on the basis vectors $(p_1 - p_0, p_2 - p_0, p_3 - p_0)$. We use an L1-loss on the difference between the predicted points and the points of the ground truth 3D bounding box. Since there are many permutations of valid 4-point corners of a bounding box, we supervise on the permutation that induces the minimum loss.

Metrics.

To make reasoning about the trade-off between IoG and IoU more quantifiable, we report the F1-score equivalent for this use case, i.e., $\text{F1}_{score} = \frac{2(IoU * IoG)}{IoU + IoG}$. We further report metrics on the dimension & pose errors, which are computed as follows:

- $err_{dim} = \text{sum}(|\mathbf{d} - \mathbf{d}_{gt}|)$, where we compute the error across all possible permutations and then choose the one with the smallest error.
- $err_{quat} = 2 \arccos(|\langle \mathbf{q}, \mathbf{q}_{gt} \rangle|)$, where $\mathbf{q}$ represents the rotational part of the pose as a quaternion. We compute the error across all possible symmetries and choose the one with the smallest error.
- $err_{center} = ||\mathbf{c} - \mathbf{c}_{gt}||_2$, where $\mathbf{c}$ is the 3D-center of the bounding box.

Results.

Table 2.2 shows how our autoregressive methods compare to the baseline parameterizations. We find that *Beam Search* achieves the best IoU, dimension & rotation error. As for the *Quantile* methods, we find that lower quantiles achieve higher IoG while sacrificing IoU and dimension error. *Quantile 0.5* offers the best tradeoff in terms of overall performance, achieving higher IoG with similar IoU and dimension error compared to *Beam Search*. Baseline models that predict box parameters directly generally performed worse since those models cannot properly capture multimodal correlations across the box parameters. The *Discrete* baseline performs the best in terms of center error, but we can see that the best autoregressive methods are only a few millimeters worse. For bounding box predictions with full rotations in SO(3), we find that an autoregressive bounding box parameterization can effectively model rotation uncertainty, achieving the lowest rotation error. We can also see that conditioning the model on known dimensions of the items in the scene increases performance in all relevant metrics (besides IoG), most notably in IoU & dimension error. Note that the dimension error is non-zero because the model is given the dimensions as an unordered set, and still needs to predict the association of each dimension tuple to the corresponding item in the scene.

Quantile and Confidence Boxes

In Section 2.4 we introduced quantile boxes as a fast approximation for confidence boxes. We showed that when $p(b|h)$ is an ordered object distribution, a quantile box with quantile q is equivalent to a confidence box with $p = 1 - q$ and should contain the true object with probability p.

While it's hard to ensure that real world objects follow an ordered distribution, we can empirically evaluate whether q confidence boxes contain the ground truth object $1 - q$ fraction of the time. To test our hypothesis, we predict quantile boxes with different q and calculate the fraction of predictions f with IoG > 0.95. In Figure 2.5, we can see that $f \approx 1 - q$ and follows a generally linear relationship. This suggests that even for general object distributions, quantile boxes can be an effective approximation for confidence boxes.

Uncertainty Measures

In Section 2.4, we introduced the uncertainty measure using quantile boxes $U_{\alpha,\beta} = 1 - IoU(b_\alpha, b_\beta)$ as a measure of the span of the confidence box interval. To evaluate the effectiveness of this uncertainty measure, we calculate the ROC AUC of using $U_{0.2,0.8}$ to predict when the IoU of the predicted box b with the ground truth box g is less than 0.25. We also measure the correlation between ground truth IoU and uncertainty using the Spearman's rank correlation r_s. We compare $U_{0.2,0.8}$ on different quantile boxes against Gaussian

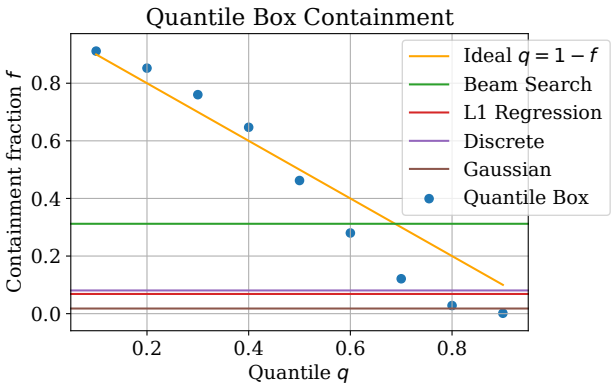

Figure 2.5: We compare fraction of predicted boxes that contain ground truth boxes f with different quantiles q and find that q-quantile boxes contain approximately $f \approx 1-q$ fraction of ground truth boxes.

Method	ROC AUC	Spearman r_s
Gaussian	0.731	-0.530
Quantile 0.2	0.897	**-0.865**
Quantile 0.5	0.878	-0.789
Quantile 0.8	**0.967**	-0.850

Table 2.3: We compare Quantile Uncertainty Measure $U_{0.2,0.8}$ with Gaussian dimension variance G, and find that $U_{0.2,0.8}$ a better predictor of ground-truth IoU compared to G as measured by ROC AUC. $U_{0.2,0.8}$ is also better correlated with ground-truth IoU compared to G as measured by Spearman r_s

dimension variance $G = \frac{\sigma_{d_x}\sigma_{d_y}\sigma_{d_z}}{\mu_{d_x}\mu_{d_y}\mu_{d_z}}$ on the Gaussian baseline. Table 2.3 shows that quantile uncertainty $U_{0.2,0.8}$ can be a better uncertainty measure than G.

2.6 Discussion

We introduced an autoregressive formulation to 3D bounding prediction that greatly expands the ability of existing architectures to express uncertainty. We showed that it can be applied to both the 3D object detection and 3D bounding-box estimation settings, and explored different ways to extract bounding box predictions from such autoregressive models. In particular, we showed how the uncertainty expressed by these models can make high confidence predictions and meaningful uncertainty estimates. We introduced a dataset that requires predicting bounding boxes with full 3D rotations, and showed that our model naturally handles this task as well. While autoregressive models are just one class of distributionally expressive models, they are not the only option for more expressive bounding box modeling. We hope that future lines of work will continue to build upon the method, dataset, and benchmarks we introduced.

Chapter 3

Latent-MaskRCNN for Instance Segmentation

3.1 Introduction

Instance segmentation is a fundamental problem in many real-world robotic systems. The goal of instance segmentation is to enumerate the objects (or *instances*) that appear in an image, specifying which pixels in the image belong to each object.

In the past few years, recent work has mostly focused on developing specialized architectures that make the instance segmentation task more amenable to deep learning. For example, *detect-then-segment* methods [24, 74, 32, 25, 7, 96], rely on a cascade of classification, regression, and filtering to first identify a *bounding box* for each instance (a related problem known as *object detection*), followed by an additional step to predict each instance's mask given its bounding box. Another example is *pixel-embedding* methods [2, 3], which optimize pixel-level auxiliary tasks, and then use a specialized clustering procedure to extract instance predictions from the dense pixel representation.

We observe that existing methods are not well equipped to deal with the inherent ambiguity that exists in the real world. We posit that this stems from a phenomenon we describe as limited *distributional expressiveness*, namely, that most instance segmentation models are designed to predict only *one* possible segmentation hypothesis (a single set of objects). From this perspective, we can think of existing instance segmentation models as point estimators; they belong to a limited function class that is not expressive enough to model the full hypothesis space. By contrast, a distributional instance segmentation model should be capable of expressing uncertainty over complex hypotheses such as, "This group of pixels might be one large object or two small ones." Making only a single prediction is limiting in terms of the accuracy attainable by high-performance autonomous systems: a robot picking application may only tolerate $< 1\%$ of errors caused by incorrect segmentation.

To overcome these limitations, we propose *distributional instance segmentation* which models a distribution over plausible hypotheses of objects. The key contributions of this

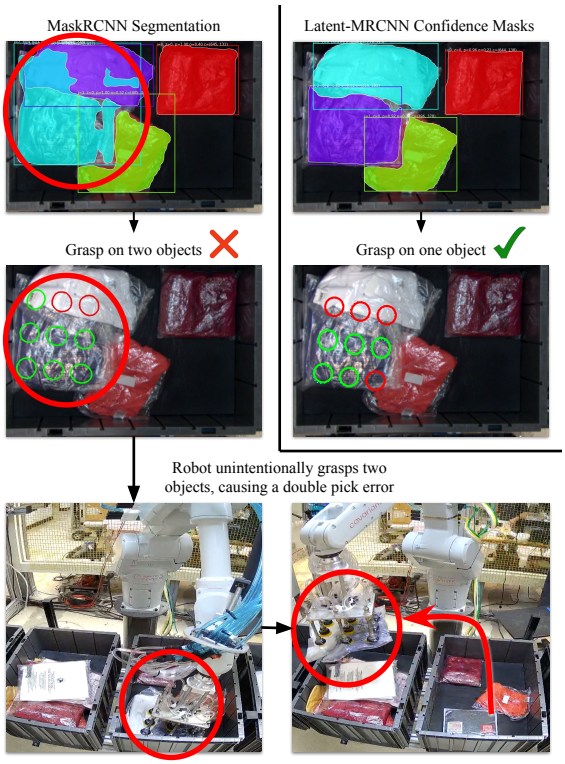

Figure 3.1: Traditional instance segmentation models such as MaskRCNN cannot model uncertainty over object masks. For robotics, this can result in critical errors such as unintentionally picking two objects. Our Latent-MRCNN can predict multiple hypotheses of object masks and use these to make high-confidence predictions, reducing the rate of double pick errors.

work are:

1. We introduce a distributional instance segmentation model using latent codes, Latent-MaskRCNN, which can predict multiple hypotheses of object masks.
2. We propose new methods for using the output of a distributional instance segmentation model. For robotic applications, we propose high-precision predictions with Confidence Masks, and we achieve high recall with Union-NMS.
3. We are releasing a dataset of over 5000 annotated images from a real-world robotics application that highlights the ambiguity in instance segmentation. We show our method achieves high performance on this dataset as well as popular driving and instance segmentation datasets.
4. On a real-world apparel picking robot, our method can significantly reduce critical errors while achieving a high level of performance (Fig. 3.1).

3.2 Related Work

Detect-then-segment methods are the most popular instance segmentation methods, and MaskRCNN belongs to this category. While they all first perform object detection and then segment each instance given its bounding box, there are some variations. For example, YOLACT [4] follows the same structure as MaskRCNN, but uses YOLO [73] as the object detector instead of FasterRCNN [74]. YOLO is very similar to FasterRCNN, making architectural changes that sacrifice some accuracy in exchange for real-time inference speed. Thus, we expect YOLACT to have the same distributional limitations as MaskRCNN. Other methods [29, 30, 62] explore how to express uncertainty during the detection step, but they consider distributions of individual boxes rather than over sets of object masks.

Mask-proposal methods [9, 89, 11] aim to circumvent bounding boxes as an intermediate representation. They are structured like FasterRCNN [74], but propose masks directly. Empirically, they do not behave much differently than MaskRCNN. Distributionally, they suffer from many of the same limitations as MaskRCNN: each proposal still models each pixel independently of the others, and they still rely on NMS to filter proposals.

Pixel-embedding [2, 3, 65, 77] methods work in a substantially different way than either of the above two families. They generally optimize some auxiliary task that encourages pixels in the same instance to have similar representations. Then they rely on a clustering-based inference procedure to extract instance predictions from their pixelwise representations. However, their performance has lagged quite far behind that of detect-then-segment methods, which has made them relatively unpopular. They can model per-pixel uncertainty in a manner similar to a naive semantic segmentation method, but this is likely insufficient for distributional expressiveness.

A number of methods explore how to express uncertainty in other structured prediction tasks. However, many of these do so by training multiple replicas of the entire model or some subset of the parameters, and modifying the training objective in a way that encourages diversity amongst the replicas [43, 27, 78, 21]. This incurs a multiplicative increase in the computational cost and memory footprint required at training time, which can be prohibitively expensive for large models. Other latent-variable formulations [42, 38, 47] offer improvements on medical *semantic* segmentation and video segmentation tasks. We find, however, that *instance* segmentation poses a richer set of challenges and has different application-specific uses.

3.3 Distributional Instance Segmentation

Distributional instance segmentation seeks to model the full distribution over object sets in an image. We pose this problem as maximum likelihood, where a model learns the conditional distribution $p(y|x)$. For a given image input x the model predicts the distribution over $y = (b, m, c)$, the bounding boxes b, masks m, and classes c.

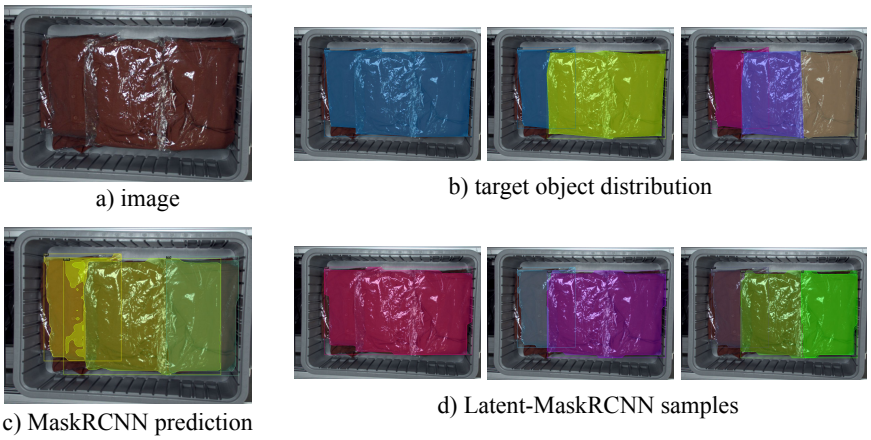

Figure 3.2: a) For this bin of items, we consider a target distribution b) of three sets of objects each with varying numbers of objects and locations. c) When MaskRCNN is trained to fit this distribution, its predictions blend objects from each of the three modes, and uncertainty in the mask head is expressed as spurious blobs on the left. d) Samples from our distributionally-expressive Latent-MaskRCNN (Section 3.4) can cleanly capture each mode of the target distribution.

If we use θ to denote the model parameters and $\mathcal{D}$ the dataset, then our training objective would be the standard maximum likelihood objective: $\max_\theta \ \mathbb{E}_{x,y\sim\mathcal{D}}[\log p_\theta(y|x)]$. In the subsequent sections we show, through two concrete examples, how MaskRCNN is unable to optimize this objective well (it can't fit the full conditional distribution $p(y|x)$).

Modeling distributions over object sets

One of the main challenges in distributional instance segmentation is modeling distributions over variable amounts of objects. Traditional approaches typically output only one set of objects through a deterministic inference procedure. However, a distributionally expressive model should be able to express uncertainty over object hypotheses by outputting multiple plausible object sets.

Consider the example illustrated in Figure 3.2, where we are not sure how many objects the image contains. We show three segmentations $y^{(1)}, y^{(2)}, y^{(3)}$ that might be plausible based on the image (Figure 3.2 (b)). If we construct a toy dataset where this image appears 3 times (once with each label), then the true conditional distribution is $p(y = y^{(j)}|x) = 1/3$, for $j = 1, 2, 3$. (This could, for instance, model human randomness in the annotation process [83]). Fitting this data requires the model to express a distribution over sets of objects.

However, MaskRCNN can only predict a single set of objects (the one that NMS produces), so the best it can do is to either choose a mode or blend the modes together. In Figure 3.2 (c), we see that it picks the mode $y^{(3)}$ at the bounding box level, and blends the

modes together at the mask level. Meanwhile, a distributional instance segmentation model would be able to fit the true conditional distribution perfectly, and express the different possibilities as 3 distinct hypotheses. In Figure 3.2 (d), we show three samples from the model we propose in Section 3.4, and observe that it correctly captures all three modes.

We find that MaskRCNN's distributional limitations restrict its ability to model the full range of possible object hypotheses. In practice, a distributional instance segmentation model is needed to capture the full range of ambiguity in the real world, which we will explore in Section 3.6.

Mask distribution expressivity

Another limitation of MaskRCNN is that its mask head cannot output an expressive distribution over mask hypotheses. Given a bounding box b_i, the mask head treats each pixel p_j within the box as independent of the others: $p(m_i) = \prod_{j=1}^{n} p(p_j)$. Samples from this distribution cannot capture any correlation between pixels.

As a concrete example, consider the image shown in Figure 3.3. In (a), there is a hand circled in blue: who does it belong to? At a first glance, it might seem like it belongs to the woman in the center of the frame, but a closer look reveals that it more likely belongs to the man on the right. In (b), we see that MaskRCNN predicts the correct bounding box for the woman, but is understandably uncertain about the hand (and even the rest of the man's arm). MaskRCNN expresses this uncertainty by outputting probabilities around 0.7 for pixels in the hand, compared to 1.0 for the rest of the woman. However, this is actually not an accurate reflection of the ambiguity that exists here: in (c), we plot one sample from the distribution in (b), and see that this is definitely not a plausible possibility. In practice, MaskRCNN would return the mode of this distribution, the ultimately-incorrect point estimate shown in (d). Meanwhile, a distributional instance segmentation model would be able to capture the correlations between pixels, and might predict a distribution over the two plausible hypotheses shown in (e) or (f): either none of the hand is part of the mask, or all of it. The sample shown in (c) would be extremely unlikely under such a distribution.

3.4 Distributional Instance Segmentation with Latent Variables

Latent Variable Formulation

How can we turn instance segmentation into distributionally expressive models, while retaining the inductive biases of existing model architectures? Drawing on prior work in variational inference [42, 40] we consider a latent-variable formulation where we incorporate latent codes in the style of a variational autoencoder. If we adopt this framework, then an instance segmentation model becomes a conditional VAE that is trained to maximize the

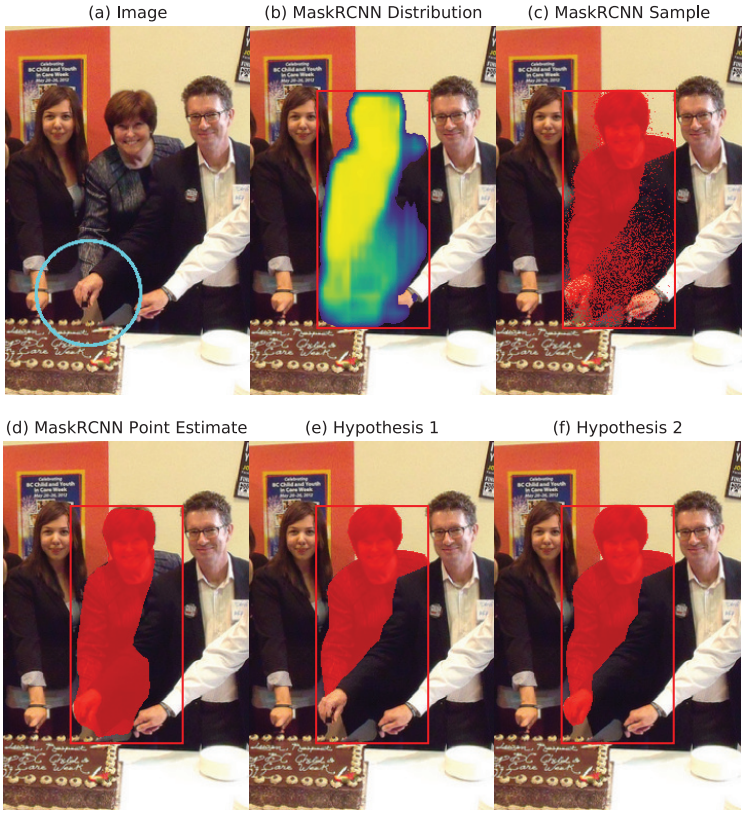

Figure 3.3: In the image shown in (a), who does the hand circled in blue belong to? (b) The pixelwise distribution predicted by MaskRCNN's mask head, which does not model the dependence between pixels. (c) A sample from this distribution, which independently samples the uncertain pixels in the hand, is not a plausible instance mask. (d) The (incorrect) point estimate that MaskRCNN would return. (e), (f) A more expressive distribution might allow us to sample plausible hypotheses like these two. In this case, (e) is actually the correct one.

evidence lower-bound:

$$\log p(y|x) \geq \mathbb{E}_{z\sim q}[\log p(y|x,z)] - D_{KL}(q||p(z|x)) \tag{3.1}$$

Typically $q(z|y,x)$ is known as the encoder, $p(y|x,z)$ as the decoder, and $p(z|x)$ as the prior, and these components are all learned to maximize the lower-bound. The decoder is essentially an instance segmentation model in the traditional sense, except that it is augmented to additionally consume a latent code z. This general technique allows us to reuse any existing instance segmentation model to implement our decoder (and train it in the same way), with only a slight modification to incorporate z as an input.

During inference, we can sample from $p(y|x)$ by sampling different latent codes $z^{(k)} \sim p(z|x)$, and decoding them into different instance predictions $y^{(k)} \sim p(y|x,z)$. This can be

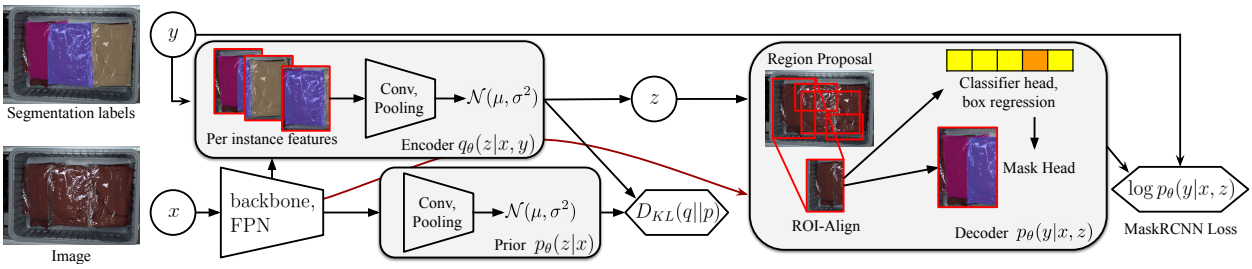

Figure 3.4: Overview of Latent-MaskRCNN: At training time, the encoder q_θ uses features extracted from the image x and labels y to sample a latent code z which is passed into the decoder. The decoder conditions on z and uses a typical MaskRCNN architecture to predict masks including region proposal, classifier, box, and mask heads. At inference time, z is sampled through the prior $p_\theta(z|x)$ which only takes the image x as input. $D_{KL}(q||p_\theta(z|x))$ ensures the prior has good coverage over the latent space.

quite powerful since we can now sample multiple structured and expressive hypotheses for a given image.

Latent-MaskRCNN

In principle, the latent-variable method can be applied to any existing instance segmentation model. In this section, we explore how it might be applied to MaskRCNN. We call the resulting model *Latent-MaskRCNN* (Fig. 3.4). We chose MaskRCNN since it is one of the most popular instance segmentation models and has served as the basis for most state-of-the-art methods in recent years.

The decoder of Latent-MaskRCNN uses the same architecture and training objective as MaskRCNN, with the main change being that it needs to incorporate latent codes. To allow them to influence as much of the prediction as possible, we want to do this relatively early in the model. We chose to inject the latent codes directly before region proposal, so that they can influence region proposal network, object detection head, and mask head. We tile the latent codes across the spatial dimensions of the image and concatenate them with the feature maps from the Feature Pyramid Network (FPN) [48]. Then we use a few convolutional layers to project the combined feature maps back down to their original channel dimensionality.

The encoder of Latent-MaskRCNN takes in an image x along with a set of ground-truth instances y, and produces a distribution over latent codes $q_\theta(z|y, x) = \mathcal{N}(\mu_\theta(y, x), \sigma^2_\theta(y, x))$. The architecture for $\mu_\theta(y, x)$ and $\sigma_\theta(y, x)$ takes inspiration from the mask head of MaskRCNN: it acts like a "reverse mask head" that operates on each ground-truth instance, and then pools features from across all instances. For each ground truth instance y_i, we extract ROI-aligned features from the FPN feature maps. Then we use a small CNN to embed each one into a single feature vector. At this point, we employ a graph neural network [88] to accumulate information from per-instance features since we need to a single latent code for

the entire image. After several graph network layers, we mean-pool across the node features and use a fully-connected layer to produce a mean and log-variance for our latent distribution. The encoder is only used at training time since it has access to the ground truth mask labels.

At inference time, we must sample latent codes from the prior to produce mask samples. This prior takes in an image x and produces a distribution over latent codes $p_\theta(z|x) = \mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x))$. We apply a few convolutional layers to the FPN feature maps, mean-pool across the spatial dimensions, and then predict a mean and log-variance using a small MLP. For all latent distributions, we use a fixed 64-dimension Gaussian with diagonal covariance.

For training, we use the encoder q_θ to sample latent codes z, which are passed to the Mask-RCNN decoder. We maximization the evidence lower bound (Equation 3.1) objective, where $-\log p(y|x, z) = \mathcal{L}_M(x, y, z)$ is the usual Mask-RCNN loss:

$$\mathcal{L}_M(x, y, z) = \mathcal{L}_{RPN} + \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{mask} \tag{3.2}$$

$D_{KL}(q||p(z|x))$ ensures the prior has good coverage over the encoder distribution. During inference, we sample latent codes from the prior (instead of from the encoder), but the decoder consumes them in the same way as during training.

We found it helpful to use a KL warm-up, as is a common practice for training VAEs [34]. The total training loss for Latent-MaskRCNN then becomes:

$$\mathcal{L}(x, y) = \mathbb{E}_{z\sim q}[\mathcal{L}_M(x, y, z)] + \beta D_{KL}(q||p(z|x)) \tag{3.3}$$

In the first part of training, we use $\beta = 0$ and increase β towards the end of training. This allows the latent code to encode useful information early on as the rest of the model is still learning; towards the end of the training, higher β pushes the latent space to be covered by the prior for better samples. For more details on our models and code, please refer to our website segm.yuxuanliu.com.

3.5 Applying Distributional Instance Segmentation

Given a distributionally-expressive segmentation model, a natural question might be, how a downstream application can consume its distributional output? Instance segmentation often occurs at the beginning of the perception pipeline, and it's not immediately clear how samples from a distributional segmentation model can be used downstream. Moreover, each application may have varying error asymmetries: failing to detect an object can be catastrophic in autonomous driving but acceptable in robotic picking, while grouping two objects as one is a critical error in robotic picking but more reasonable in driving applications. In this section, we show how a single Latent-MaskRCNN model can be used flexibly across a number of applications with different requirements (Fig. 3.5).

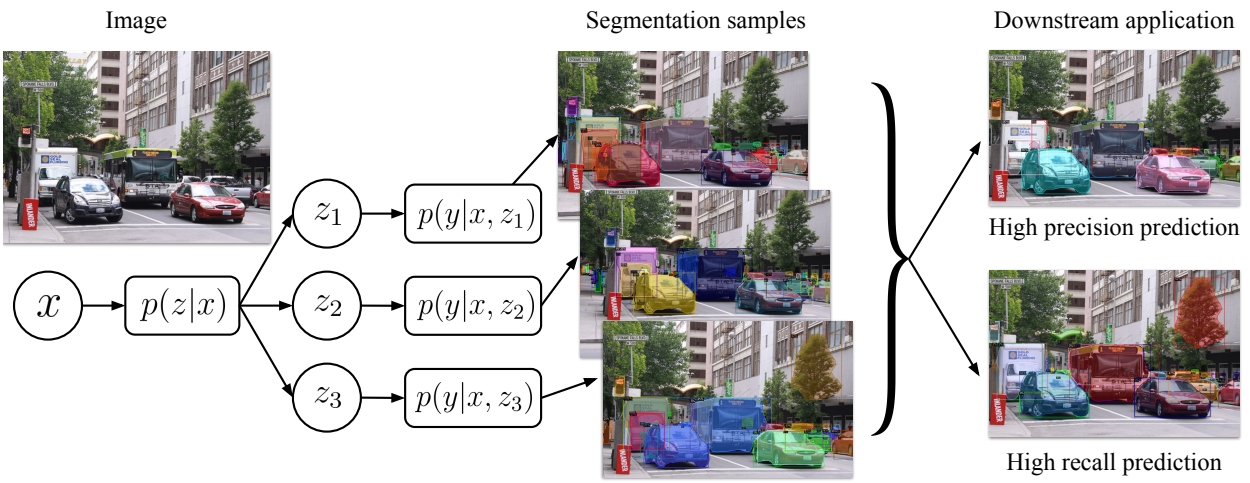

Figure 3.5: At inference time, the encoder q_θ is discarded, and latent variables z_i are sampled from the image x conditioned prior $p_\theta(z|x)$. Each latent is decoded using $p_\theta(y|x, z_i)$ into a set of masks, which can be used for our high precision or recall predictions depending on the application.

High-Precision with p-Confidence Masks

In some applications, it can be very costly to make *under-segmentation* errors, when an instance's mask is predicted to be larger than it actually is. For example, consider a robotic manipulation application, where the robot must pick objects one at a time and feed them into a sortation process. If the model undersegments an instance, it may inadvertently pick multiple objects, which can be an expensive error for the downstream application. How can we ensure that these errors don't occur?

Suppose we draw several samples from Latent-MaskRCNN. If two pixels belong to the same instance mask in many samples, then we can be reasonably confident that they actually do belong to the same ground-truth instance. Drawing on this intuition, we can then compute a *p-confidence mask*, consisting of pixels that are all likely to be contained in a single ground-truth instance.

For a given confidence requirement p, we define a confidence mask c_p as a mask that is fully contained in a ground truth mask m with probability at least p: $\mathbb{P}(c_p \subseteq m) \geq p$. Using Latent-MaskRCNN, we can approximate this probability as:

$$\mathbb{P}(c_p \subseteq m) = \mathbb{E}_{p(m|x)}[\mathbb{1}\{c_p \subseteq m\}] \approx \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}\{c_p \subseteq m_i\}$$

In the finite sample regime, $\hat{c}_p$ is an empirical confidence mask if it is contained within a sampled mask for p fraction of the samples.

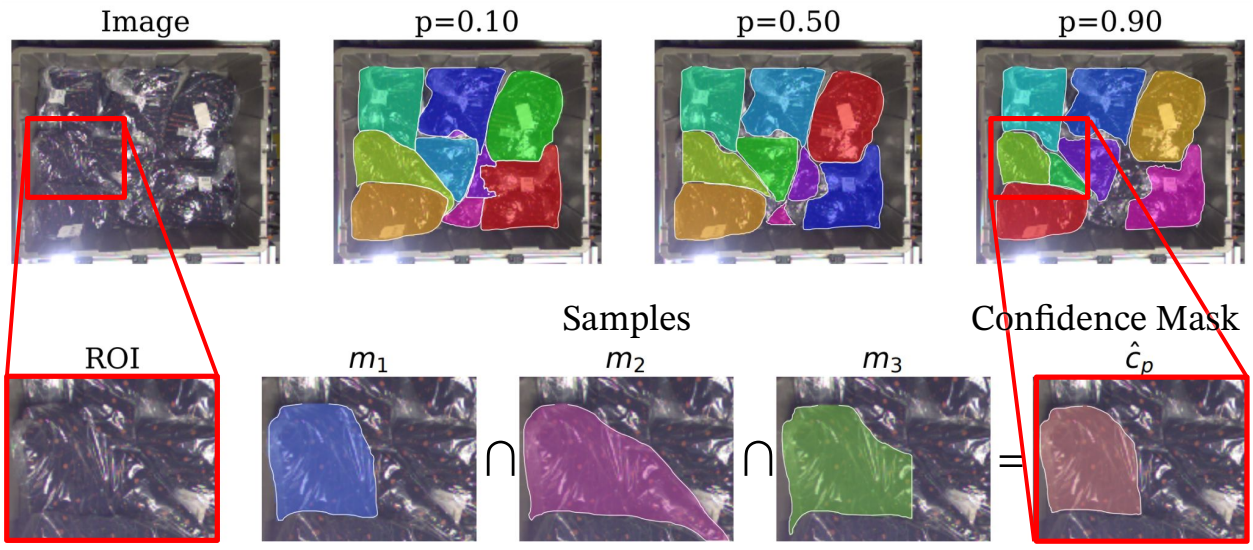

Figure 3.6: Top: Confidence Mask predictions with different p. Notice that as the confidence requirement p increases, single objects can be split into two, ambiguous object extents are reduced, and uncertain objects are eliminated entirely. Bottom: Constructing an empirical confidence mask $\hat{c}_p$ by taking intersection of samples m_1, m_2, m_3. When an object's extent is uncertain, a high confidence mask prediction will only consist of pixels that are highly likely to be contained within an object as determined by the samples.

Now consider any subset of masks I consisting of one mask m_j from at least kp different samples. If we take the intersection of all of the masks in I, $\hat{c}_p = \bigcap_{m_j \in I} m_j$, then this intersection mask must be contained in each of the masks used in the intersection $\hat{c}_p \subseteq m_j$. Therefore we have:

$$\frac{1}{k} \sum_{m_j \in I} \mathbb{1}\{\hat{c}_p \subseteq m_j\} = \frac{|I|}{k} \geq p$$

and $\hat{c}_p$ is an empirical confidence mask by construction.

Figure 3.6 illustrates confidence mask predictions for different p. Notice as the confidence requirement increases, the unconfident extents of the masks shrink, and some uncertain masks are eliminated. We can also see how constructing confidence masks via intersection leads to high confidence region predictions.

Scoring Confidence Masks

Since each confidence mask is an intersection of masks $c_p = \bigcap_I m_j$, how should we assign the score of a confidence mask prediction? One intuitive approach might be to take the average of the score s_i of each mask in the intersection: $\frac{1}{|I|} \sum s_j$. However, a confidence mask is not an average of masks but rather an intersection.

To formulate a better score for our confidence mask prediction, consider two scenarios. In the first scenario, the model is very confident about an object's mask so it predicts roughly the same mask in every sample. The resulting confidence mask c_p has high IoU with each of the masks used in the sample m_j. On the other hand, consider an unconfident prediction where the object's mask varies significantly across samples. Here, the confidence mask c_p represents a small but confident region of the object whose extent is highly uncertain. The resulting IoU between c_p and each m_j will be smaller than when the model is confident and masks are not varying across samples.

We score each confidence mask as the mean score-weighted IoU between the predicted mask c_p and every mask used in the intersection:

$$s_{c_p} = \frac{1}{|I|} \sum_{m_j \in I} s_j \frac{|c_p \cap m_j|}{|c_p \cup m_j|}$$

When s_{c_p} is large, this indicates that c_p is a confident intersection of masks with very similar IoU. On the other hand, a small s_{c_p} indicates that c_p has low score or IoU with its samples and likely does not capture the full extent of the object well.

To predict a set of confidence masks for an image, we iteratively select the highest scoring confidence mask, excluding the pixels of all of the confidence masks predicted so far. This algorithm greedily approximates the maximum scoring confidence mask selection optimization.

High-Recall with Union-NMS

Other applications might be concerned about *over-segmentation*, the complement of under-segmentation. For example, in autonomous driving, failing to identify a pedestrian, or predicting them to be smaller than they actually are, can lead to a catastrophic error.

To make high-recall predictions with Latent-MaskRCNN, we use a procedure called *Union-NMS*. We first sample multiple segmentations from the model, and run NMS on the predicted masks. It checks if any two masks m_i, m_j have IoU greater than some threshold, and then discards the lower-scoring one. Suppose that some mask m_i remains after we perform NMS. Then Union-NMS returns the union of m_i with every mask that it suppressed, achieving higher recall by incorporating masks that would have otherwise been ignored. We describe the Union-NMS algorithm in full detail in Appendix B.4.

Vanilla Prediction with the Prior Mean

Some applications may not have any specific performance requirements or may have strict inference time requirements. In these cases, a point estimate can be sufficient. With Latent-MaskRCNN, we can achieve this by always decoding the mean of the prior: $z = \mu_\theta(x)$ where $\mu_\theta(x)$ is the mean of the prior $p_\theta(z|x)$ (for Gaussian $p_\theta(z|x)$, it is also the mode of the distribution). We found that this scheme typically matches or yields a small improvement over

MaskRCNN predictions, suggesting that Latent-MaskRCNN strictly increases the expressiveness of MaskRCNN and no performance is lost by using a more expressive distribution.

3.6 Experiments

We conducted experiments seeking to answer the following questions:

1. Can Latent-MaskRCNN with Confidence Masks make high-precision predictions across a variety of datasets?
2. Can Union-NMS make high-recall predictions?
3. Can Latent-MaskRCNN reduce critical double pick errors in robotic picking applications?

Datasets

To help us answer these questions, we compared MaskRCNN and Latent-MaskRCNN across several datasets, each with its own set of challenges.

COCO [49]: This large dataset is the standard benchmark for instance segmentation. There are many object categories and a huge variety in image composition.

Cityscapes [14]: A real-world dataset from an autonomous driving application. Although it is smaller and more specialized than COCO, it is still a popular benchmark for instance segmentation. One notable challenge is that there are many background instances that are still important to segment (e.g. pedestrians), but the limited image resolution can introduce some uncertainty.

Apparel-5k: We collected this dataset of roughly 5000 images from a robot picking application. We use 4198 images in the training set and 463 in the validation set. There is only one object category, but the images exhibit a lot of inherent ambiguity due to complex occlusions, lighting, transparency, etc. We are releasing this dataset on our website segm.yuxuanliu.com for the broader community to build upon our work.

For each dataset, we trained both MaskRCNN and Latent-MaskRCNN on 8 GPUs using MaskRCNN's released hyperparameters and training schedules. We use the same publicly available train/val splits for all experiments and datasets. We used a Resnet-50 backbone [31], initialized from pretrained-Imagenet [16] weights (for COCO) or pretrained COCO weights (for other datasets). Inference with MaskRCNN can take 80-100ms depending on the number of objects, and inference with Latent-MaskRCNN can take 500-1000ms depending on the number of samples and objects.

Qualitative Evaluation of Samples

In this section, we qualitatively explore what kinds of uncertainty Latent-MaskRCNN can express. In Figure 3.7, we visualize samples from the model on images from various datasets,

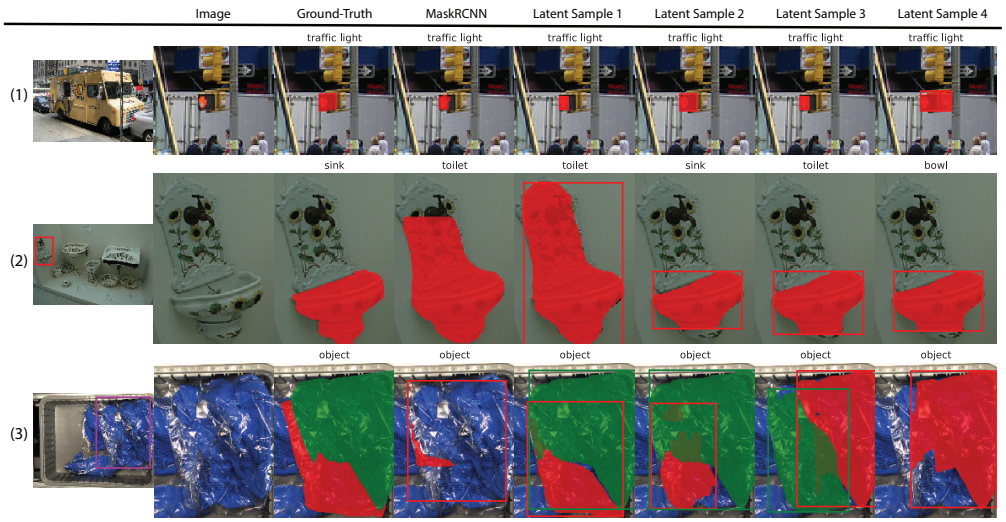

Figure 3.7: Each row corresponds to a single image. Columns 1-2 show the original image and a zoomed-in version. The remaining columns show instance masks, and the word above the image is the class of the instance. Column 3 is the ground truth, column 4 is MaskRCNN's prediction, and columns 5-8 are samples from Latent-MaskRCNN.

and observe that it does capture several distinct types of ambiguity:

Category confusion: Latent-MaskRCNN can express meaningful uncertainty in the classification head. In row 2, MaskRCNN confidently classifies this sink as a toilet, while different samples from Latent-MaskRCNN classify it as toilet, sink, and bowl.

Category imprecision: Even when the class of an instance is obvious, there still may be ambiguity in how the category is defined. For example, in row 1, both MaskRCNN and Latent-MaskRCNN are (correctly) confident that this instance is a traffic light. However, depending on how you define exactly what constitutes the extent of the traffic light, the instance mask may look very different. Latent-MaskRCNN samples a wide range of plausible possibilities, but MaskRCNN picks a single (in this case incorrect) mode.

Object and mask ambiguity: Latent-MaskRCNN lets us sample from a distribution over sets of objects. For example, in row 3, we see that the samples contain different numbers of objects, variety in bounding boxes, and variety in instance masks. Even though none of the samples are perfect, they are all plausible and all markedly better than MaskRCNN. In row 2, we see examples where Latent-MaskRCNN's hypotheses express uncertainty in both mask and bounding box, while MaskRCNN picks a single mode.

Evaluating p-Confidence Masks

In Section 3.5, we introduced high precision predictions with p-Confidence masks to address the problem of under-segmentation. In those cases, we care that predictions have high

Table 3.1: Evaluation of MaskRCNN and Latent-MaskRCNN across various datasets and metrics.

	COCO			Cityscapes			Apparel-5k		
Method	MR@HP	AR	mAP	MR@HP	AR	mAP	MR@HP	AR	mAP
MaskRCNN	20.0	66.0	35.0	25.3	55.1	**35.8**	23.6	39.9	26.9
Latent Union NMS	7.4	**72.3**	26.5	17.7	**57.5**	33.6	13.6	**61.4**	34.1
Latent Confidence Mask	**22.0**	48.1	30.5	**28.0**	48.6	34.1	**42.4**	41.8	**35.1**
Latent Prior Mean	19.5	65.8	**35.3**	25.7	53.8	35.0	26.9	49.4	34.3

Intersection-over-Prediction: $\text{IoP}(m_i, g) = \frac{|m_i \cap g|}{|m_i|}$. When IoP is high, errors due to under-segmentation are less likely to occur.

When evaluating models in this regime, we need to trade off precision (in terms of IoP) with recall (to avoid degenerate solutions). To do this, we consider the *max recall at high precision* (MR@HP):

$$\text{MR@HP} = \frac{1}{|p| \cdot |\tau|} \sum_{p_i \in p, \tau_j \in \tau} \max_{t:\text{Precision}(\tau_j) \geq p_i} \text{Recall}(t, \tau_j)$$

For a given precision threshold p_i and IoP threshold τ_j, we can compute the max recall that each model achieves (or zero, if it never achieves precision p_i). The MR@HP metric is the average of these recalls, over a range of precision threshold p and IoP thresholds τ. For high precision use-cases, we care about performance at high values of these thresholds, therefore we use $p = \tau = [0.75, 0.8, 0.85, 0.9, 0.95]$.

In Table 3.1, we evaluated Latent-MaskRCNN using both the prior-mean scheme from Section 3.5 as well as confidence masks with a confidence level $p = 0.9$. Across all three datasets, we find that latent confidence masks yield the best performance in terms of MR@HP. As for mAP, we find that Latent Prior Mean can match if not exceed the performance of MaskRCNN on all three datasets. On the challenging Apparel-5k dataset, we find that Latent Confidence Mask and Latent Prior Mean significantly outperform MaskRCNN in terms of MR@HP and mAP. Overall, we find that Latent-MaskRCNN is a strict improvement over MaskRCNN by matching overall detection performance in terms of mAP and offering the best high-precision performance in terms of MR@HP.

Evaluating Union NMS

For the over-segmentation problem, we introduced the Union NMS method in Section 3.5. In such cases, we care that we have high recall (that we detect every instance that exists), and that each mask prediction has high *IoG* (intersection-over-ground-truth): $\text{IoG}(m_i, g) = \frac{|m_i \cap g|}{|g|}$. To capture both of these considerations, we consider the average-recall (AR) [49] using IoG. This measures both recall while also penalizing over-segmentation (are there any predicted masks that are too small).

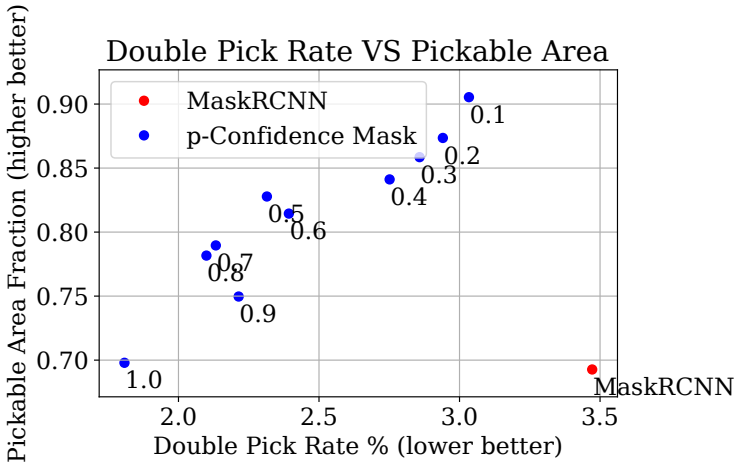

Figure 3.8: Latent confidence masks achieve lower double pick rates and generally more pickable area compared to MaskRCNN.

We evaluated Latent-MaskRCNN using both its prior-mean (Section 3.5) and Union-NMS. In Table 3.1, we show that the prior-mean predictions are similar to MaskRCNN on all three datasets, while Union-NMS achieves substantially higher AR (using IoG). This suggests that Latent-MaskRCNN with Union NMS can more effectively cover different modes of uncertainty, for high-recall applications.

Can confidence masks reduce double pick errors on Apparel-5K?

In a robotic picking application, it is costly for the robot to pick up two items accidentally, thinking it had only picked one item since it affects inventory counts and downstream orders. For the Apparel-5k dataset, we can estimate the double pick rate of a model's segmentation prediction by approximating the robot's gripper as a circle with a fixed radius in pixel space. Then, we randomly sample circles on the image and count the number of circles D that land within one predicted mask but more than one ground truth mask. We divide this by the number of circles N that land within one predicted mask, to arrive at the estimated double pick rate $R = \frac{D}{N}$. Empirically we find that this simulated double pick rate is correlated with double pick rates on a real robot.

Another metric that we are concerned with in industrial robot picking is pickable area, the amount of visible surfaces that the robot can pick from. A model that predicts higher, more accurate pickable areas enables the robot to have more flexibility in its grasping strategy. To this end, we compute the area of all the predicted masks over the area of all the ground truth masks, as the fraction of pickable area available.

We compare MaskRCNN and Latent-MaskRCNN with varying p-confidence masks in

Figure 3.8. We find that Latent-MaskRCNN outperforms MaskRCNN in fraction of pickable area and double pick rate in all cases. Moreover, the tunable parameter p in Latent confidence masks allows for application-specific tradeoffs between double pick rate and pickable area. Higher values of p tend to correspond to lower double pick rate and less pickable area, as the confidence requirement for each prediction is increased. With traditional MaskRCNN, only one double pick rate and pickable area fraction is realizable since no tunable knob exists.

Can confidence masks reduce double pick errors on a real-world apparel-picking robot?

To evaluate whether our dataset evaluation translates to real-robot performance, we compare MaskRCNN and Latent-MaskRCNN on an apparel-picking robot. We use an ABB1300 with a 9-cup suction gripper to pick apparel items in polybags between two totes (Fig. 3.1). The robot uses two overhead camera systems to perform instance segmentation and then grasp point generation. The grasp points are optimized to land as many suction cups as possible on a single object detected by the segmentation model.

For our evaluation, we only change which segmentation model is used while holding other parts of the system constant, including hardware, object set, and grasp point generation. Each segmentation model is trained on the same Apparel-5k dataset. We run each model with several hundred grasps and record the number of double picks, grasps that unintentionally pick two objects. In an industrial warehouse application, these double picks are very costly errors since they result in incorrect inventory counts and cause errors in downstream sortation and order fulfillment systems. A typical high automation warehouse can tolerate at most 1% double pick rate before the robot is causing more problems than it solves.

We also measure the average number of sealed cups on a grasped item, since the suction holding force is proportional to the number of sealed cups. Grasps that use less sealed cups tend to result in more dropped objects, which leads to jams, lost inventory, and costly human intervention. A robotic system can reduce double pick rate by shrinking object mask sizes, chopping up bigger masks into smaller ones, or only using a single small suction cup. However, all of these approaches indiscriminately reduce the suction holding force on all items, whereas our approach will be conservative only when ambiguity is present.

Table 3.2 reports the results of our apparel-picking experiments. We find that Latent-MaskRCNN with 0.9-Confidence mask significantly reduces the double pick rate. This validates our simulated findings on Apparel-5K in Section 3.6. Moreover, Latent-MaskRCNN achieves slightly better average number of sealed cups, suggesting that suction stability was not sacrificed. This suggests that our method can make high-confidence predictions and make the appropriate trade-offs in the face of uncertainty.

Table 3.2: Apparel-picking robot evaluation. * indicates a statistically significant difference

Method	Double Pick Rate	Average Sealed Cups
MaskRCNN	4.40%*	4.76
Latent-MRCNN	**0.82**%*	**4.91**

3.7 Discussion

We proposed a new family of models that builds on top of existing instance segmentation models by using latent variables to achieve more distributional expressiveness. Latent-MaskRCNN can express a wide range of uncertainty where existing instance segmentation models often fall short. We can leverage uncertainty expressed by the model using Confidence Masks and Union-NMS to achieve high precision and high recall respectively. These methods demonstrate strong performance across robotics, autonomous driving, and general object datasets. On a real apparel-picking robot, we find that our model can significantly reduce the rate of critical errors while maintaining high performance. Finally, we have highlighted the importance of distributional expressiveness and hope that future work in instance segmentation can continue to build on top of our work and datasets.

Chapter 4

Self-Supervised Instance Segmentation

4.1 Introduction

Instance segmentation is often the basis of many robotic applications, including object grasping, manipulation, and placement. Given an image, the goal of instance segmentation is to predict the set of pixels that belong to each object. After objects are detected, a robot can use segmentation masks to execute more generalizable grasping and manipulation policies. A robust robotic application must be able to recognize thousands of new objects on an ongoing basis. Instance segmentation also offers clear semantics for application logic such as "only grasp on one object (not two)" and "count the number of objects in this bin."

Much of the recent advances in instance segmentation [12, 32, 44, 51] assume that large-scale labeled datasets of objects from known classes are available [49, 18]. This assumption, however, does not hold for many robotics applications that must handle a constant stream of new objects. Collecting and annotating such a dataset is also costly and expensive. How can a robot learn to segment a diverse range of objects with only limited labeled data?

Consider a grasping robot that uses suction cups to pick objects from one bin and place them into another, such as in a warehouse automation application. Suppose we train an instance segmentation model on a small set of labeled data, and use this model with a grasping policy to generate grasps that land on the predicted masks, achieving decent performance on the initial object set. Every week, the warehouse introduces new SKUs of objects for the robot to pick; over time, the segmentation model struggles to recognize the new objects and the grasping performance degrades. To ensure good grasping performance, we would need to continually label data from the warehouse, incurring an recurring cost. Can we improve the segmentation performance without expensive human annotation?

An object can be defined as a contiguous group of pixels that move together [8]. When the robot successfully grasps an object, the pixels of that object are removed from the scene. The mask of the grasped object can then be inferred from the grasp location, before image,

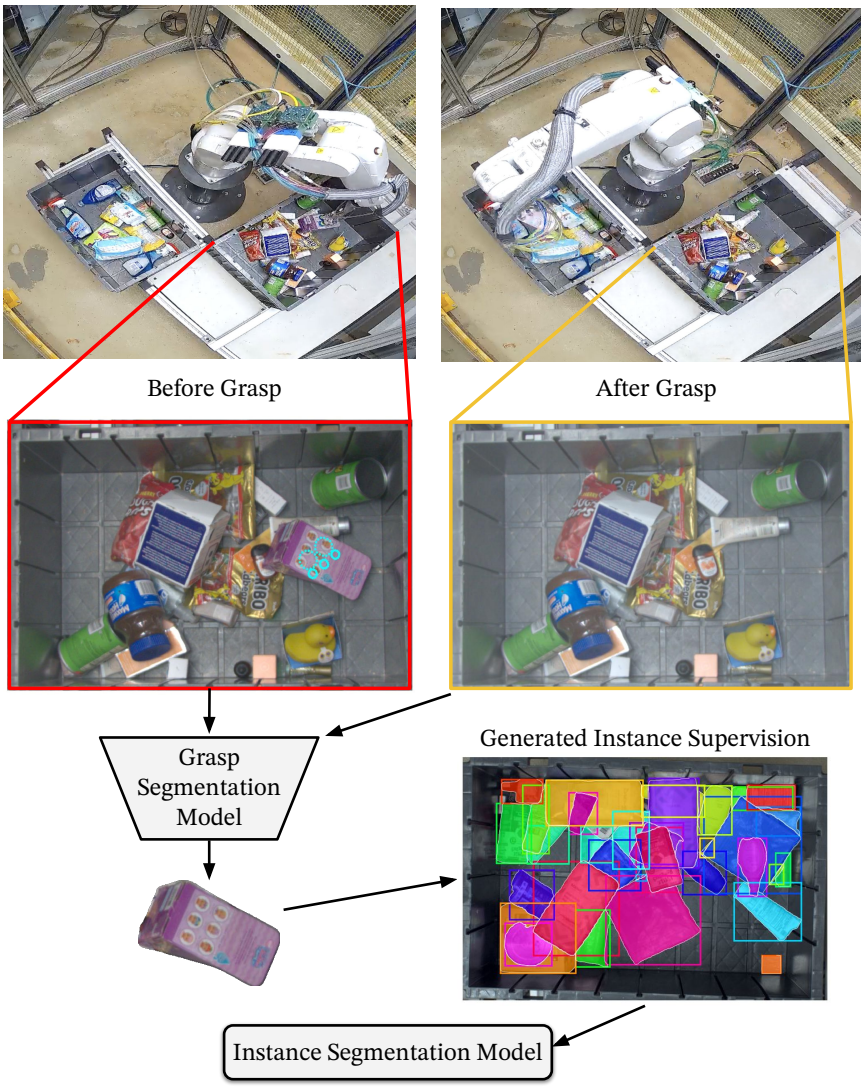

Figure 4.1: Overview of our method: the grasp segmentation model takes before and after images of the grasp to predict the mask of the grasped object. We use the grasp segmentation model to get object masks for thousands of grasps in a self-supervised manner. These grasped objects can be "cut" and "pasted" to generate diverse training supervision for the instance segmentation model.

and after image of the scene. Leveraging this insight, we propose a grasp segmentation model that predicts the mask of the grasped object. A grasp segmentation model differs from an instance segmentation model in that it only needs to predict one object (the grasped one), and has additional information (before/after images and the grasp location) that enables it to generalize better even when trained on a small dataset.

Traditional methods of segmenting grasp interactions often use image and background subtraction which are not robust to occlusions, reflections, and other objects moving [68]. We find that our learned model overcomes these limitations and can robustly segment grasped objects, even when other objects in the scene have shifted during grasping. In our experiments, we show that our grasp segmentation model is significantly more accurate at segmenting grasped objects than traditional image subtraction approaches.

Once we have a grasp segmentation model, we can run this on thousands of unlabeled grasps to get self-supervised object masks. To reduce noise, we use a suction gauge to keep only successful grasps and propose an uncertainty-aware filtering method to keep high-accuracy masks. With these filtered object masks, we can then "cut" objects from their original scenes and "paste" them into new scenes with random augmentations to generate instance supervision [1, 95]. We combine this generation scheme with inpainting to generate a diverse set of photo-realistic scenes with infinite variation of scale, rotation, and occlusion. This allows our self-supervised robotic system to continually learn to segment new objects, and improve on known objects, without human annotation. Figure 4.1 shows an overview of our self-supervised instance segmentation method.

The key contributions of this method are as follows:

1. We propose a self-supervised robotic grasping system that can continually learn and improve its instance segmentation, on new and known objects, without human annotation.
2. Our novel grasp segmentation model uses before and after grasp images to segment grasped objects with 5x less error than traditional approaches, while being robust to occlusions, reflections, and other object moving in the scene.
3. We introduce a "cut-and-paste" and inpaint method to generate supervision for instance segmentation models that outperforms the same model trained with 10x the amount of labeled data.
4. On a robotic grasping task, we show that models trained with our method can reduce the rate of grasping failures by over 3x compared to an image subtraction baseline.

4.2 Related work

Instance Segmentation

There has been a significant amount of recent advances in the field of instance segmentation, with many approaches focusing on supervised learning on large datasets such as COCO [49]. Detect-then-segment is among the earliest learned approaches that use a two-stage object detection and segmentation architecture [32]. More recently, single-stage models that use transformer attention mechanisms, such as Mask2Former, have demonstrated strong performance [12, 10, 7, 55]. However, all of these methods rely on the availability of large-scale annotated datasets, which may be costly to obtain for robotics applications that must

handle a constant stream of new objects. In addition, our method is agnostic to the instance segmentation model, so we can leverage all past and future advances in instance segmentation models.

Self-Supervised Segmentation

To address this issue, there have been several approaches that propose self-supervised or semi-supervised methods for instance segmentation. Cut and paste methods have been shown to improve instance segmentation performance [1, 95]. Prior work [68, 56] proposed self-supervised approaches that use image subtraction on before and after grasps to provide instance segmentation supervision. However, masks recovered from naive image subtraction are imperfect, resulting in worse performance at higher IOU thresholds, which is insufficient for high-performing robotic applications. Optical flow methods can also infer contiguous groups of pixels that move together as the robot is pushing them around [17, 8, 92]. This may be an impractical approach if continuous high-bandwidth video is not available from the robot camera, or if the robot is expected to be grasping objects instead of pushing them in a production environment.

Moving Object Detection

Another class of methods can detect moving objects in video sequences, such as traffic and surveillance footage. These methods can use a background subtraction approach by modeling a static background scene with a mixture of Gaussians [39, 99]. Pixels that are noticeably different from the background model, as determined at the pixel level or with local features [26], are segmented as moving objects. More recently, convolutional neural networks have been applied to learn this background subtraction with 2D and 3D convolutions [5, 36]. Moving object detection models, however, are not directly applicable for grasp segmentation since they segment all objects that have moved instead of the one that was grasped. If the robot moves an adjacent object to the grasped object, both objects would be segmented as one under moving object detection, providing incorrect instance segmentation supervision. Moreover, methods that learn a background model will have limited data since the background scene changes with every grasp.

Representation Learning

Other approaches have proposed learning object *representations* for robotics instead of instance segmentation directly. In Grasp2Vec [37], representations of the object and scene are learned to satisfy arithmetic consistency. These representations can then be used to learn policies that manipulate and grasp objects. Similarly, pixel-wise descriptors can be learned for each object using a contrastive loss [19]. While learned object representations can be used for robot manipulation, they have yet to be proven effective for learning instance segmentation. The clear semantics of instance segmentation may be desirable for some robotic

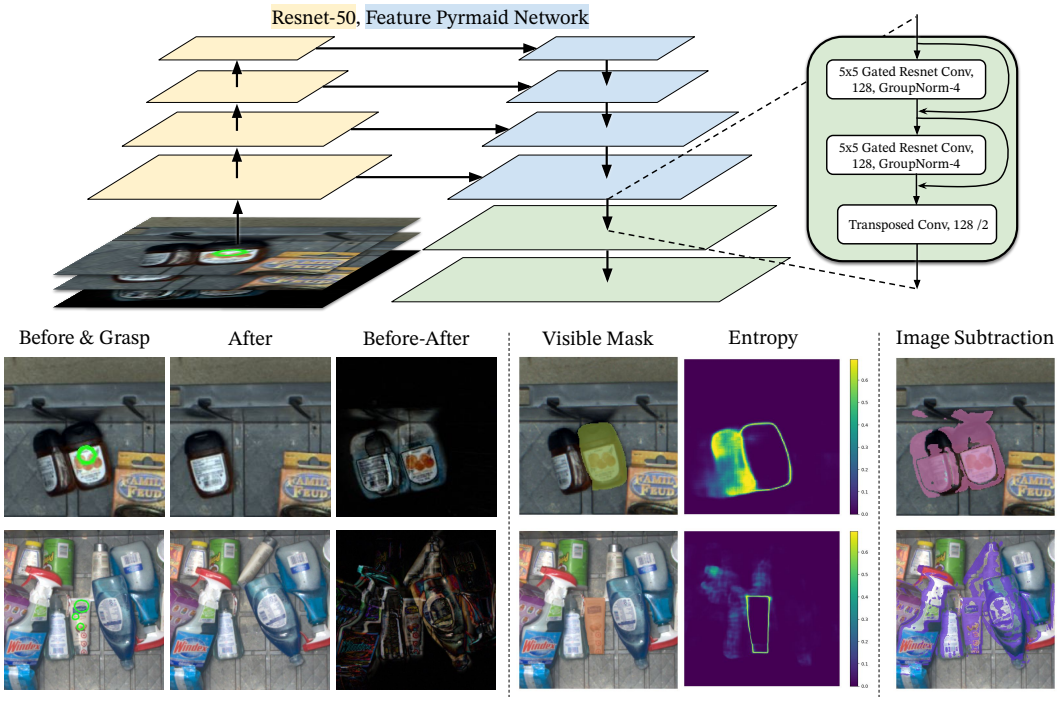

Figure 4.2: Our grasp segmentation model takes before, after, grasp, and subtraction images as input to predict the mask of the grasped object. We build upon a Resnet-50 backbone with a Feature Pyramid Network (FPN) and two additional learned upsampling convolution blocks to predict masks at the image resolution. Traditional image subtraction methods (on the right) fail to properly handle reflections and other objects moving in the scene. Our learned approach can correctly segment the grasped object and express uncertainty via the entropy of the prediction.

applications such as counting the number of objects, or ensuring that grasps only occur on a single object.

4.3 Instance Segmentation by Grasping

Our method uses grasp interactions to collect segmentation data for objects in a self-supervised manner. We propose a grasp segmentation model that can robustly segment grasped objects from before and after grasp images. We show how our model can predict masks that are robust to occlusions, reflections, and other objects in the scene changing. Then, by combining our grasp segmentation model with our uncertainty-aware filtering method, we can collect a dataset of grasped object masks from unlabeled grasp images.

Grasp Segmentation Model

Our grasp segmentation model is inspired by prior work that uses grasp interactions to infer object masks and representations based on sets of pixels that move together [68, 37]. Given a before grasp image i_b, an after grasp image i_a, and a grasp mask g, our model predicts the visible mask of the grasped object m_v. This is similar to image subtraction which subtracts the before and after images $|i_b - i_a| > t$ and uses a threshold t to determine the object mask. However, image subtraction is a very brittle segmentation method with very few tunable parameters that limit its robustness to occlusions, reflections, and other moving objects.

Our model, on the other hand, uses a neural network to predict both the visible m_v and amodal mask m_a (including occlusions) [45] of the grasped object. By explicitly reasoning for occlusions, we can filter out objects that would have been partially visible and providing incorrect supervision for downstream instance segmentation. A trained model will also learn to ignore inputs, such as reflections and other moving objects, that are not relevant to segmenting the grasped object.

We base the grasp segmentation model architecture on a Resnet-50 [31] with a Feature Pyramid Network (FPN) [48], which has been shown to be effective at segmenting objects at multiple scales. To initialize the model with features that are amenable for object detection, we load weights from a Mask-RCNN [32] pre-trained on COCO [49]. We include image subtraction $i_b - i_a$ as an input to the model since it provides a good inductive bias for which pixels have changed. Altogether, we concatenate the inputs $[i_b, i_a, i_b - i_a, g]$ along the channel dimension before passing them into the Resnet-50 backbone. Since the input dimensions are different than the usual RGB inputs for Resnet, we randomly initialize the first layer of the Resnet-50 while initializing all other weights from the pre-trained Mask-RCNN.

To make predictions at the original input resolution, we take the highest resolution, stride-4 feature layer from the FPN, and apply a series of convolutions and transposed convolutions to upsample the features. We use two blocks of gated residual convolutions followed by a transposed convolution for upsampling. Similar to regular residual blocks $y = x + C_1(x)$, gated residual blocks use an additional learned gating operation $y = x + \sigma(C_2(x))C_1(x)$ where σ is the sigmoid function and C are learned convolution operators, like in LSTMs [35]. For our application, this enables the model to easily ignore irrelevant features such as using the reflective object features to mask out the predicted object mask. Finally, we make a 2-channel prediction $\hat{m}_v$, $\hat{m}_a$ corresponding to the visible and amodal object masks respectively. Figure 4.2 illustrates our model architecture overview along with sample inputs and predictions on how our model can be more robust than image subtraction.

We supervise all predictions using a binary cross entropy loss with weighting w

$$CE(\hat{m}, m, w) = \frac{1}{n} \sum_{i=1}^{n} w_i (m_i \log \hat{m}_i + (1 - m_i) \log(1 - \hat{m}_i)) \tag{4.1}$$

Due to the imbalanced nature of the classification task, we use multiple weightings to ensure the neural network gives the appropriate attention to the relevant pixels. The first two weights, $w^{(1)} = m, w^{(2)} = \neg m$ provide a balanced weighting. Since it's important for the

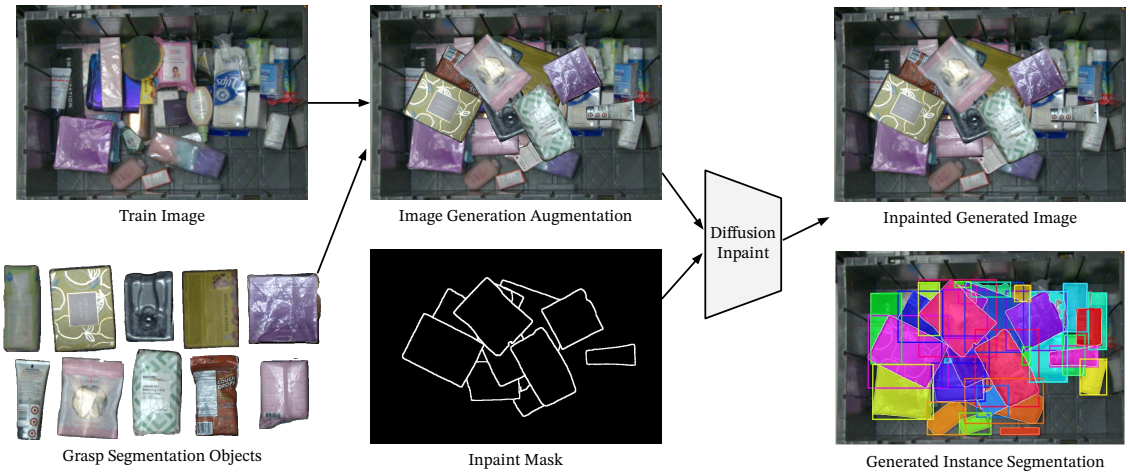

Figure 4.3: By applying the grasp segmentation model to thousands of grasps, we can collect a large dataset of grasp segmentation objects in a self-supervised manner. Then we can generate instance segmentation supervision by taking a random subset of segmented grasped objects and "pasting" them onto any training image. We apply augmentations such as rotation, scale, and offset to generate cluttered scenes of objects. Then, we use a pre-trained diffusion inpaint model to smooth out the pasted object boundaries to be more photorealistic. The resulting inpainted image and pasted masks can be used to train any off-the-shelf instance segmentation model.

pixels near the mask boundary to be accurate, we use $w^{(3)\ldots(6)} = maxpool(m, k)$ with kernel sizes $k = (11, 51, 101, 201)$. This enlarges the region around the object and focuses the model's loss to predict an accurate object boundary.

We train with the total loss

$$L(\hat{m}, m) = \sum_{j=1}^{6} CE(\hat{m}, m, w^{(j)}) \tag{4.2}$$

using Adam with learning rate 5e-6 with a batch size of 8 until convergence. We train the grasp segmentation model using supervised learning on a small dataset of 100-200 labeled grasp images. To improve generalization, we use augmentations during training including random crops, resizes, blurs, and color adjustments. Even with such a small training set, the model can generalize to provide accurate segmentation masks for thousands of grasps on new objects.

Supervising Instance Segmentation

Once our grasp segmentation model is trained, how can we learn an instance segmentation model? We propose a 4-step approach: first, we use the grasp segmentation model to collect

and filter object masks across a large variety of grasped objects. Then, we "cut-and-paste" object masks using augmentations to generate cluttered scenes of objects. We use inpainting on object boundaries to reduce pasting artifacts, and use the inpainted images to train a state-of-the-art instance segmentation model. Figure 4.3 illustrates an overview of our approach.

Collecting Accurate Object Masks:

First, we use the grasp segmentation model on unlabeled grasp image pairs to predict the mask of the grasped object. To collect a reliable set of unoccluded object crops, we apply a several filters on the prediction. We compute the sums of visible and amodal masks $S_v = \sum \hat{m}_v, S_a = \sum \hat{m}_a$, and compare the ratio $\frac{S_v}{S_a} > t_{occ}$ with a threshold. We use a threshold $t_{occ} = 0.95$ to filter out objects the model predicts as not fully visible.

The grasp segmentation model could also predict masks that are wrong or uncertain. To filter out uncertain masks, we can use entropy thresholding where

$$\mathcal{H}(\hat{m}) = -\hat{m} \log \hat{m} - (1 - \hat{m}) \log(1 - \hat{m}) \tag{4.3}$$

is the binary entropy. We can threshold average relative entropy of the visible mask prediction $\frac{\sum \mathcal{H}(\hat{m}_v)}{\sum \hat{m}_v} < t_{ent}$ to filter out uncertain predictions. We found $t_{ent} = 0.1$ to provide a reasonable trade-off between precision and recall.

Finally, predicting discontinuous masks can indicate that more than one object was detected or the model is predicting spurious blobs. To avoid this kind of prediction error, we use OpenCV's findContours function to find contiguous groups of contours. Then we count the number of contours with at least 1000 pixels and only keep objects with 1-2 contours.

Image Generation Augmentation:

Once we have filtered out uncertain predictions and obtained a set of accurate object masks, we can use these masks to generate supervision for instance segmentation. We use object masks to "cut" images of the objects and "paste" them onto annotated images to generate instance supervision [1, 95].

To train the model to be robust to a diverse set of objects, we randomize select 0-25 objects to be pasted on a random training image. We apply different geometric transformations to the objects such as randomly rotating up to 360 degrees, scaling between 0.75x-1.25x, and randomly selecting a position. This can help the model learn to better handle variations in the object orientations, positions, and occlusions in the images. With a small amount of labeled data, we can augment with self-supervised grasp segmentation to generate a large dataset of instance segmentation supervision

In-Painting With Diffusion:

While naive cut-and-paste can generate diverse supervision, the augmented images may contain artifacts such as wrong object boundaries and unrealistic shadows. These artifacts

can lead a model to learn features that are not present in natural images. To bring the generated images closer to the natural image space, we use a pre-trained Stable Diffusion in-painting model that is trained on a large-scale image dataset [75].

First, we take the visible boundary of the objects that are pasted and dilate the boundary by 5 pixels on all sides to cover the region on the object boundary. We use this mask as the in-paint mask input to a pre-trained Stable Diffusion in-painting model and run the denoising process for 4 steps on the masked region. This provides a more photo-realistic boundary between the background and the object.

Instance Segmentation Model:

Since our method directly generates instance segmentation supervision, it's compatible with any off-the-shelf instance segmentation model. This enables greater flexibility of our method since we can take advantage of any past and future advances in model architecture. We chose a state-of-the-art model, Mask2Former [12], as the main instance segmentation model for all of our experiments. Mask2Former uses a multi-scale, masked-attention transformer decoder with learned queries and a Hungarian matching loss on the masks. We use the official public implementation of Mask2Former with a Resnet-50 as the backbone and default parameters (100 queries).

4.4 Evaluating Grasp Segmentation

First, we'll evaluate the performance of our grasp segmentation model compared to traditional methods such as image subtraction. Since grasp segmentation provides object masks for instance segmentation supervision, the accuracy of the grasp segmentation will influence the performance of downstream instance segmentation.

Grasp Data Collection

For our grasping experiments, we use an ABB-1200 robot with a 5 suction cup end-effector as shown in Figure 4.1. The robot uses RGB-D cameras to plan pick and place motions that cycle the objects between two bins. We record the image (640x960 pixels) before the grasp i_b, the image after the grasp i_a, and the grasp mask g corresponding to the pixels of the active suction cups. After grasping, we use suction gauges of each cup to determine whether the object was indeed picked and whether the grasp was successful. While this data collection method requires an instance segmentation model with decent performance to generate grasps, we use the suction gauge to keep only successful grasps in the dataset which can bootstrap even a poorly performing model.

We collect 110k grasp image pairs across a variety of training and test objects. We label 1k images from the training set and 6k images from the test objects with instance segmentation labels. This enables us to train and evaluate both grasp and instance segmentation. The test objects and the training objects are distinct and have no overlap.

Background and Image Subtraction

In order to evaluate the performance of our grasp segmentation model, we will compare it to traditional methods such as image and background subtraction [68, 39, 99, 26]. Image subtraction is a simple method that subtracts the before-grasp image from the after-grasp image to obtain the grasp mask. This method assumes that changed pixels correspond to the grasped object. To make the image subtraction more robust, we apply greyscale and Gaussian blur before the subtraction, $G(I) = GaussianBlur(Greyscale(I))$. Then we use the thresholded difference as the segmentation mask

$$\hat{m} = |G(I_a) - G(I_b)| > t$$

We also consider background subtraction methods such as mixture of gaussians MOG [39], MOG2 [99] and Local SVD binary pattern (LSBP) [26]. MOG models the background with a mixture of gaussians while LSBP uses local image features to detect changes. With all image and background subtraction methods, we can apply the same OpenCV contour approximation from Section 4.3 as a filter.

Evaluation Results

We train the Grasp Segmentation model on a subset of 100 and 200 grasps from the training set, until convergence as described in Section 4.3. Then we evaluate on the test set of hold-out object grasps, using mean intersection over union (mIOU) as the evaluation metric. We also report the relative error rate, which is the number of incorrectly predicted pixels divided by the number of pixels in the ground truth mask. For methods that use filtering described in Section 4.3, we calculate the recall, which is the percentage of mask predictions not filtered out. When using filtering, the mIOU and Error rates are calculated only on the kept data after filtering. If a method makes no pixel predictions for a grasp, this prediction will be filtered as well.

How Do Our Methods Compare With Baselines?

The results of our evaluation are summarized in Table 4.1. All grasp segmentation models perform better than image and background subtraction baselines (Subtract, MOG, MOG2, LSBP). As shown in Figure 4.2, these traditional approaches are unable to account for other objects moving in the scene and will confuse pixels that change with similar intensity.

How Does Performance Vary With Training Set Size?

We can see that the grasp segmentation model with filtering trained on 200 grasps, Grasp-200-Filter, performs the best. Training with 100 grasps and filtering (Grasp-100-Filter) achieves slightly worse mIOU, error rate, and less than half the recall of Grasp-200-Filter). This suggests that increasing the amount of training data improves the accuracy and the number of scenes the model can confidently segment.

Table 4.1: Grasp Segmentation Evaluation

Method	mIOU	Error	Recall
Subtract	51.1	396%	98.6
Subtract-Filter	61.6	44.5%	50.4
MOG	51.8	333%	98.3
MOG-Filter	59.9	47.1%	65.4
MOG2	32.4	1289%	99.8
MOG2-Filter	47.3	970%	42.0
LSBP	37.7	1050%	99.8
LSBP-Filter	55.6	359%	44.0
Grasp-100-NoAug	78.3	31.5%	100
Grasp-100	78.7	22.6%	99.8
Grasp-100-NoAug-Filter	80.4	26.9%	92.3
Grasp-200	86.8	14.3%	99.6
Grasp-100-Filter	91.1	9.05%	29.4
Grasp-200-Filter	**92.6**	**7.63%**	63.9

What's The Impact Of Filtering And Augmentations?

Filtering improves the mIOU and error rates while decreasing recall for all methods. We also see that not using data augmentation, Grasp-100-NoAug-Filter, achieves lower accuracy metrics but higher recall. This suggests that fewer objects are filtered out and the model is confidently wrong since it has overfit to the small training set. Data augmentation is important to training a robust grasp segmentation model.

4.5 Evaluating Instance Segmentation

To evaluate instance segmentation performance, we use the Mask2Former model with R50 backbone and default hyperparameters as described in Section 4.3. We train and evaluate on the dataset from Section 4.4 and use early stopping to prevent overfitting. We use 100 labeled training images for most evaluations, while benchmarking on 1000 labeled training images as a reference. We evaluate instance segmentation models using standard metrics, such as overall Average Precision (AP), IOU thresholded AP (AP@0.5, AP@0.5), and object size breakdowns (AP^L, AP^M).

For Paste methods, we take the corresponding model trained in Section 4.4 to generate object crops. We apply the same inference and filtering for each method on the 110k unlabeled grasp dataset. Using the filtered object crops, we then paste them randomly onto the training set following the procedure described in Section 4.3. As an ablation, we also paste from training image crops to evaluate our augmentation scheme without grasp data

Table 4.2: Instance Segmentation Evaluation

Method	Paste From	Labeled Images	AP	AP@0.5	AP@0.75	AP^L	AP^M
Mask2Former-100	None	100	49.30	70.75	51.47	55.34	41.70
Single-Object-100	None	100	22.10	24.05	23.22	27.03	15.48
Paste-Subtract	Subtract-Filter	100	62.11	80.13	67.01	69.97	52.83
Paste-Subtract-Robust	Subtract-Filter	100	65.07	86.24	72.51	72.97	53.10
Paste-Train	Train Objects	100	66.28	84.22	70.54	73.27	55.88
Paste-Grasp	Grasp-100	100	69.68	87.37	75.37	77.45	58.47
Paste-Grasp-Filter	Grasp-100-Filter	100	69.22	86.15	74.62	77.63	56.72
Paste-Grasp-Robust	Grasp-100-Filter	100	63.02	85.33	70.63	70.47	51.54
Paste-Grasp-Inpaint	Grasp-100 + Inpaint	100	**72.33**	**88.25**	**78.11**	**80.68**	59.40
Mask2Former-1000	None	1000	70.08	85.69	74.34	76.12	61.48

(Paste-Train).

AP@0.5 and AP@0.75 measure how well the model can detect objects that are at least half and three-quarters overlapping with the predicted mask, respectively. AP measures the overall performance of the instance segmentation model averaged over multiple IOU Thresholds. AP^L and AP^M measure the performance of the model on small and medium objects, respectively. (AP^S was not meaningful as there were very few small objects).

Baseline Methods

Single Object Supervision:

One intuitive way to use grasp segmentation data for instance segmentation is to supervise only on the grasped object. With the Mask2Former loss, this can be done by using only the predicted grasp segmentation in the Hungarian matching and ignoring the "no object" loss term. We combine single-object supervision from the Grasp-100 model with full supervision from the training set at a 50:50 ratio (Single-Object-100).

Robust Set Loss:

To overcome errors in noisy masks generated by image subtraction, prior work [68] proposed a robust set loss, which requires the predicted mask to be within only a margin of the ground truth mask. Given a predicted mask, the robust set loss will use a discrete optimization to find the closest target mask that is within some IOU margin with the ground truth. We use the publicly available implementation of robust set loss with IOU threshold 0.7 and replace the Mask2Former mask loss with the robust set loss.

Evaluation Results

How Do Baseline Methods Compare?

Table 4.2 summarizes the results of our evaluation. Training with single object supervision, Single-Object-100, actually results in worse performance than just training on the full supervision dataset (Mask2Former-100). We suspect that there isn't enough negative supervision from the single-object data since there are only labels for a single object in each image. We find that pasting with objects from filtered image subtraction (Paste-Subtract), provides a performance improvement over just supervised learning (Mask2Former-100). Adding the robust set loss (Paste-Subtract-Robust) further improves the segmentation performance which is consistent with [68]. Pasting from the labeled training objects, Paste-Train, outperforms Paste-Subtract-Robust on AP. This suggests that the accuracy of cropped objects in the training set outweighs the diversity of the less correct crops from image subtraction. Moreover, pasting provides a strong augmentation for learning rotational, scale, and occlusion invariant instance segmentation.

Does Robust Set Loss Always Improve Performance?

Using the robust set loss with object masks from the more accurate, grasp segmentation model (Paste-Grasp-Robust) actually hurts performance, unlike in the image subtraction case. This suggests that the robust set loss is beneficial only when there is a significant amount of error in the grasp segmentation, such as with image subtraction's 44.5% error. With the 9.05% error rates of our grasp segmentation model, supervising with a normal cross entropy loss is better.

What's The Impact Of Filtering?

Using object masks from our learned Grasp Segmentation model (Paste-Grasp) can further improve instance segmentation performance. Paste-Grasp and Paste-Grasp-Filter have similar results which suggest there is some trade-off between the error of the object mask and how many objects are filtered out. Recall from Table 4.1 that Grasp-100 achieves 22.6% error with 99.8 recall, while Grasp-100-Filter has 9.05% error with 29.4 recall. The non-filtered model (Paste-Grasp) performs slightly better which suggests that the diversity of the objects seen outweighs the accuracy of the object segmentation here.

How Does Inpainting Affect Performance?

Finally, we see that using inpainting with the grasp segmentation model (Paste-Grasp-Inpaint), achieves the best performance, outperforming a Mask2Former model trained on 10x the amount of labeled images on all but one metric. This suggests that inpainting the pasted objects can produce more realistic supervision for learning instance segmentation features.

Table 4.3: Robot Evaluation Results

Method	Grasp Error Rate
Paste-Subtract-Robust	29.02%
Mask2Former-1000	8.78%
Paste-Grasp-Inpaint	**8.22%**

4.6 Robot Grasping Evaluation

Experimental Setup

To evaluate the effectiveness of our method in a real robotic application, we use our trained instance segmentation models to detect objects for grasping. Using the same grasping setup from Section 4.4 and the models trained in Section 4.5, we compare grasping success with different instance segmentation models. A grasp attempt is counted as a success if it picks up exactly one object and places it into the other bin without dropping. If the predicted mask is not an actual object, or contains multiple objects, this would be a grasp failure. The grasping system will try to land as many cups as possible on a detected object mask and plan a collision-free path to reach that object. By keeping all other parts of the grasping system constant and only changing the segmentation model, we can isolate the effects of the segmentation model on grasp performance.

Evaluation Results

We compare the top segmentation models from each method: supervised learning (Mask2Former-1000), robust image subtraction (Paste-Subtract-Robust), and our grasp segmentation model with inpainting (Paste-Grasp-Inpaint). We perform 700 grasps on the same object set for each segmentation model. The results of our robotic grasping evaluation are shown in Table 4.3.

Overall we found the grasping evaluation to be consistent with the instance segmentation evaluation in Section 4.5. Paste-Subtract-Robust performs the worst with most of its grasp failures due to predicting masks that don't belong to any object, such as reflections in the wall or empty areas of the bin. Our Paste-Grasp-Inpaint model achieves the lowest grasp error rate that is over 3x better than the image subtraction baseline and comparable to a model trained with 10x the amount of labeled data, Mask2Former-1000.

Failure Analysis

Figure 4.4 highlights three failure categories caused by incorrect instance segmentation that we will discuss here.

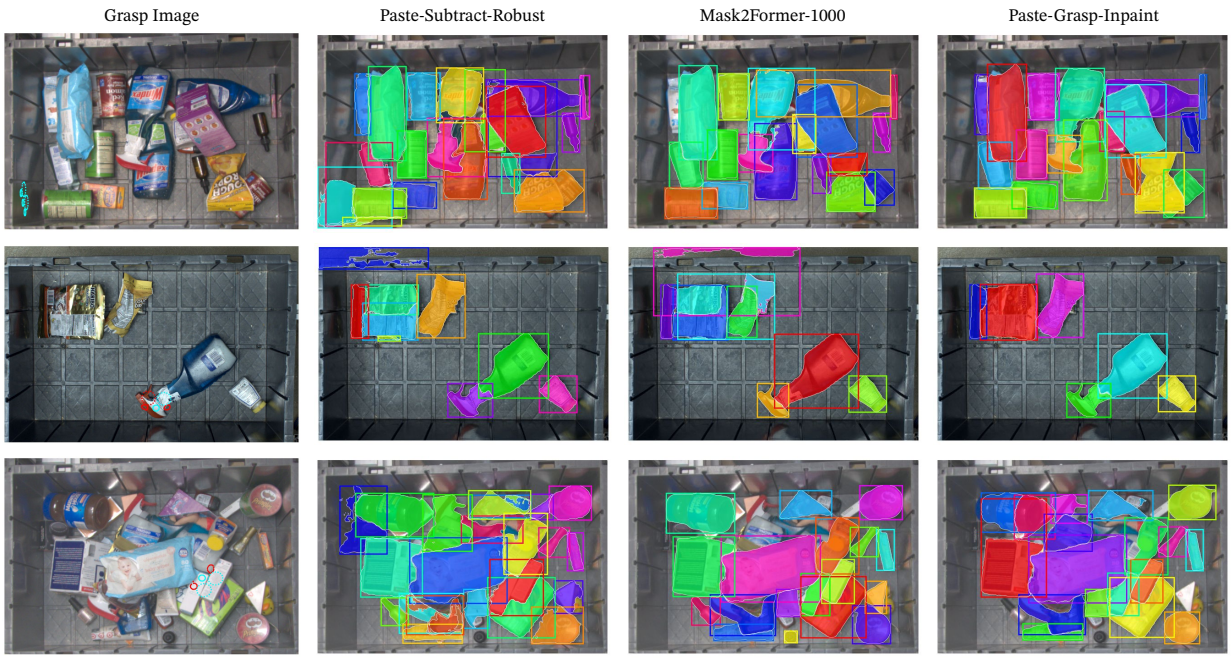

Figure 4.4: Visualizations of different types of grasp failures from real robot execution caused by wrong segmentation. The first column shows the executed grasp with active cups colored in blue and inactive cups colored in red. Columns 2-4 show the predicted segmentation from each of the 3 models used in the evaluation. The first row shows an example of a grasp on a non-object that was incorrectly predicted by the segmentation model. The second row is a grasp on an unstable part of the object that leads to a drop; the segmentation model splits the object into two masks, causing an unstable grasp to be executed. The third row is a grasp on two objects; this is caused by the segmentation model grouping two objects into one mask.

Grasp On Non-Objects:

In the first row, the robot grasps on the reflection in the bottom left corner of the bin, which fails since it's not on an object. Grasping on non-objects such as reflections and empty areas of the bin is a common failure of the Paste-Subtract-Robust model. We suspect image subtraction may have incorrectly segmented the reflection or bin as an object, which then provided misguided supervision to the instance segmentation model. Our learned model, on the other hand, is robust to these errors and does not make the same mistakes.

Splitting Objects Into Two:

In the second row, an unstable grasp on the top part of the bottle leads to the robot dropping the item. Here the segmentation model incorrectly split one object into two masks, resulting

in a grasp that is not centered on the object. All three models fail to segment the bottle correctly, however our learned Paste-Grasp-Inpaint model has the least amount of errors on other objects.

Grouping Two Object As One:

Finally, when the segmentation model incorrectly groups two objects as one, this can lead to a grasp on two objects. Only the Paste-Subtract-Robust model suffers from this failure case, which suggests that image subtraction incorrectly grouped two objects that moved as one object. This incorrectly grouped object was then used to supervise the model, leading to this grasp error. This can cause inventory errors and inconsistencies in warehouse applications Failures between the supervised and self-supervised models were similar, ranging from grasps on two object with similar textures to cases where the segmentation was correct but the grasp sampled could not physically seal on the item. This suggests that our instance segmentation model trained with self-supervised grasp masks can effective for robotic grasping applications, without requiring a large, costly labeled dataset.

4.7 Conclusion

In this work, we proposed a novel method for instance segmentation that utilizes self-supervised grasp images to generate object masks for training. We showed that our grasp segmentation model can accurately detect objects with high mIOU and low error rate, even when trained on a small number of labeled images. We then use the grasp object masks to train an instance segmentation model with a inpainting augmentation method, which outperforms a model trained with 10x the amount of labeled data. We have also shown that our method leads to improved grasping performance in a real-world robotic application. This work highlights the potential of using self-supervised grasp images for learning instance segmentation models, and opens up new possibilities for training such models in a wide range of robotic applications.

Chapter 5

Conclusion

In this thesis, we have addressed the challenges associated with perception for real-world robotic applications by introducing novel models and self-supervised learning methods. We proposed three methods: (1) Autoregressive bounding box prediction, (2) Latent-MaskRCNN for instance segmentation, and (3) Self-supervised instance segmentation. These methods not only capture and handle uncertainty but also leverage diverse data collected in real robotic applications without the need for costly human annotation.

By incorporating uncertainty estimation and self-supervised learning, we have demonstrated significant improvements in various robotic systems, reducing critical errors and enhancing overall performance. Our contributions pave the way for more robust, adaptable, and high-performing robotic systems that excel in complex and dynamic environments, addressing the unique challenges posed by real-world robotics and bridging the gap between AI research and practical robotic applications.

5.1 Future Work

While our proposed methods have demonstrated considerable improvements in real-world robotic applications, there are still several avenues for future research.

Instance Segmentation

The field of instance segmentation has seen exciting developments recently, especially with the advent of text-prompted segmentation techniques. These methods hold considerable promise for improving the adaptability and efficiency of robotic systems. Here, we propose several directions for future work based on recent advances in this area.

Text-Prompted Segmentation for Rapid Adaptation

The "Segment Anything Model" (SAM) [41] presents a compelling approach to text-prompted segmentation, enabling models to detect and segment objects based on textual prompts. If

a robot needs to detect a new edge case such as a broken product, it can take many weeks to define the semantics of that class to train annotators, collect and annotate data, and train a new model to detect this case. However with a general segmentation model such as SAM, robotic systems can detect new object classes within minutes and quickly adapt to new objects or edge cases. For future work, we propose to explore this text-prompted segmentation approach in the context of real-world robotics. This could involve developing methods to effectively integrate text prompts into the robot's perception pipeline and expanding the support of such models on scenes a robot would see.

Iterative Image and Text Prompting with Human Feedback

The "Segment Everything Everywhere All at Once" (SEEM) [100] paper offers an intriguing concept of combining visual and text prompting in an iterative manner, supported by human feedback. While a model's initial prediction through text prompting alone may not result in the correct prediction, by expanding the prompt space to include bounding boxes, scribbles, outlines, and even dots, humans can arrive at the right prediction through visual prompting. This expands the traditional notion of a "prompt engineer" to beyond just text to include visual modes of prompting, enabling the model to refine its predictions and better align with human expectations.

Building on this concept, we also propose to investigate methods for integrating iterative prompting and human feedback into robotic systems. This could involve developing interfaces for humans to provide feedback to the robot, and algorithms for the robot to interpret this feedback and adjust its behaviors accordingly. Such an approach could significantly improve the robot's ability to handle challenging tasks and adapt to new situations beyond just at the segmentation level.

Modeling Uncertainty in Segmentation

Both the "Segment Anything" and our Latent-MaskRCNN methods acknowledge the inherent ambiguity in segmentation tasks and attempt to model this uncertainty. For instance, the "Segment Anything" approach predicts three possible masks for each object, allowing for multiple plausible segmentations. Segmentation has natural ambiguity that depends on the application-specific use-case. For example, should a water bottle be segmented as a bottle and a cap, or as one object? This may depend on if the robot is trying to open the bottle or just manipulate it as one object.

We propose to further explore this aspect of modeling uncertainty in segmentation tasks. This could involve developing new methods to quantify and express uncertainty in segmentation predictions, as well as investigating ways to incorporate human feedback into this process. For instance, when a robot is uncertain about it's predictions, it could actively prompt the human for feedback, which can then be incorporated into the models future predictions. By enabling more flexible and uncertainty-aware predictions, we can enhance

the ability of robotic systems to handle ambiguous situations and improve their overall performance and reliability.

Self-Supervised 3D Representations for Robotics

In robotics large annotated dataset are often scarce, and labelling them is expensive. In this thesis, we've already explored how we can use self-supervised learning methods to learn instance segmentation from grasping interactions. However, the richness of the 3D world that robots operate in suggests a broader question: can we extend these principles to learn rich 3D object representations?

Camera on Arm Data Collection

A promising direction for future work involves mounting cameras on the robot's end effector. By capturing RBGD images as the robot is manipulating various objects, we can collect a wealth of data for learning 3D representations. Moreover, using the robot's joint angles and calibration data, we can recover pose and correspondence supervision across multiple viewpoints. The resulting data can be leveraged to learn a wide range of object representations outlined below.

Dense Descriptors

The question of what constitutes the "right" object representation for manipulation is critical in robotics. The "Dense Object Nets" paper [19] presents an object representation that satisfies several desirable properties: task-agnostic, applicable to both rigid and non-rigid objects, leverages 3D vision, and is learned from self-supervision. However, these descriptors are intended to work across a single object class. Building upon these ideas, future work could investigate how to generalize descriptors and affordances across novel objects, enabling multi-object descriptor learning. This will enable a robot's manipulation policy learned on one object to generalize quickly to new objects.

Neural Rendering

Another promising avenue involves the incorporation of Neural Radiance Fields (NeRFs) [61, 91] into the learning process. By integrating NeRFs, we can learn object descriptors that are not only useful for recognition and segmentation, but also for rendering objects from new viewpoints. This could enable the generation of a diverse range of synthetic data, enriching the robot's learning environment and enhancing its capacity to generalize to novel situations.

In line with the concept of Text-to-NeRF [81], future work could also investigate the possibility of grounding 3D object representations in text. This approach could allow robots to synthesize and learn from new 3D objects described solely in text, significantly expanding the range of objects the robot can handle. By capitalizing on the wealth of data available

from robot interactions and integrating methods like NeRFs and text-grounded learning, we can equip robots with a more powerful representation of objects they need to manipulate.

Sim2Real and Diffusion Models

Another promising direction for addressing data scarcity in robotics lies in the potential of simulation-to-reality (sim-to-real) transfer. By leveraging the virtually infinite data generation capabilities of simulated environments we can augment real-world data to build even more robust models.

One of the earliest and most influential approaches to sim-to-real transfer is domain randomization [85]. This technique introduces variations in the simulated environment's parameters, enabling the model to learn generalizable features that remain robust to changing conditions. More recent advancements have explored the use of diffusion models to generate photo-realistic supervision for perception models. [93] By using diffusion inpainting, we can position an object or robot into an infinite variation of photorealistic backgrounds.

In future work, the integration of simulation environments with photo-realistic diffusion techniques could provide a powerful framework for generating diverse and realistic data for robotic training. By creating a wide range of simulated scenarios and rendering them with high photorealism, we can provide a rich training resource that prepares robots for the vast diversity of the real world without costly annotation.

Suction Modelling

While a significant portion of the robotics literature focuses on finger-based grippers, suction-based manipulation has emerged as an industry-preferred alternative due to its versatile grasping capability. Yet, comprehensive and versatile modeling of suction-based manipulation remains an open research question, particularly suction stability across a diverse set of cup types and object materials.

Existing works, such as Dex-Net 3.0 [57] and methods from the Amazon Robotics Challenge [94], have made valuable strides in modeling suction. Dex-Net 3.0, for example, introduces a compliant suction contact model that calculates the seal quality between the suction cup and the local target surface. Zeng et al. [94] also use human annotation to learn an affordance model for suction-based grasping. Yet, such models are only the beginning, and much remains to be understood about the nuances of suction-based manipulation.

We hope that future work can continue to delve deeper into the modeling of suction-based manipulation and develop more sophisticated models that account for the diversity of real-world scenarios. This includes the variability of object materials, from rigid cardboard to deformable plastics, and the different suction cup types, each with its own unique properties and performance characteristics. By leveraging the wealth of data that can be collected during robotic operations, these models can continuously learn and improve, leading to more effective and reliable suction-based manipulation in a wide range of industrial applications.

Modeling Uncertainty

The progress we have made in this thesis with models like Latent-MaskRCNN and Autoregressive bounding boxes has shed light on the potential benefits of modeling uncertainty in both 2D and 3D object recognition. In future work, we should strive to build models that can make distributional predictions and reason about uncertainty. This will be essential in building any safety-critical or high-performing application.

The very nature of real-world robotics involves dealing with complex, dynamic, and often unpredictable environments. Given this, it is crucial that our AI models have the capacity to understand and express their limitations – to know what they don't know. This is not just limited to robotics, but extends to AI in general, including text-based models such as ChatGPT, which has been known to make confident hallucinations [67]. By making AI's uncertainty explicit, we allow humans to better understand the system's limitations and make more informed decisions about when and how to trust the AI's outputs.

5.2 Conclusion

In conclusion, this thesis navigates the intersection of AI and robotics, introducing novel models that handle real-world ambiguity and utilize self-supervised learning, addressing the gaps in current robotics applications. The future encompasses large-scale, interactive segmentation models, enhanced 3D representations, refined suction modeling, sim-to-real techniques, and models that reason about uncertainty more effectively. We hope this work is a stepping stone towards making real-world industrial robotics more reliable and efficient, setting the stage for exciting and promising research directions.

Bibliography

[1] Relja Arandjelovic and Andrew Zisserman. "Object Discovery with a Copy-Pasting GAN". In: *CoRR* abs/1905.11369 (2019). arXiv: 1905.11369.

[2] Anurag Arnab and Philip H. S. Torr. "Pixelwise Instance Segmentation With a Dynamically Instantiated Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[3] Min Bai and Raquel Urtasun. "Deep watershed transform for instance segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2017, pp. 5221–5229.

[4] Daniel Bolya et al. "YOLACT: Real-time Instance Segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.* (2019).

[5] Marc Braham and Marc Van Droogenbroeck. "Deep background subtraction with scene-specific convolutional neural networks". In: *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2016, pp. 1–4. DOI: 10.1109/IWSSIP.2016.7502717.

[6] Anthony Brohan et al. "RT-1: Robotics Transformer for Real-World Control at Scale". In: *arXiv preprint arXiv:2212.06817*. 2022.

[7] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *ECCV*. 2020.

[8] Honglin Chen et al. "Unsupervised Segmentation in Real-World Images via Spelke Object Inference". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022.

[9] Xinlei Chen et al. "Tensormask: A foundation for dense object segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.* (2019).

[10] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. "Per-Pixel Classification is Not All You Need for Semantic Segmentation". In: 2021.

[11] Bowen Cheng et al. "Masked-attention Mask Transformer for Universal Image Segmentation". In: 2022.

[12] Bowen Cheng et al. "Masked-attention Mask Transformer for Universal Image Segmentation". In: 2022.

[13] Jiwoong Choi et al. "Gaussian YOLOv3: An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 502–511.

[14] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2016, pp. 3213–3223.

[15] Angela Dai et al. "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2017.

[16] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Ieee. 2009, pp. 248–255.

[17] Andreas Eitel, Nico Hauff, and Wolfram Burgard. "Self-supervised Transfer Learning for Instance Segmentation through Physical Interaction". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China, 2019.

[18] Mark Everingham et al. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* (2010).

[19] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. "Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation". In: *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 373–385.

[20] Markus Freitag and Yaser Al-Onaizan. "Beam Search Strategies for Neural Machine Translation". In: *Proceedings of the First Workshop on Neural Machine Translation*. Vancouver: Association for Computational Linguistics, Aug. 2017, pp. 56–60. DOI: 10.18653/v1/W17-3207. URL: https://aclanthology.org/W17-3207.

[21] Bin-Bin Gao et al. "Deep label distribution learning with label ambiguity". In: *IEEE Transactions on Image Processing* 26.6 (2017), pp. 2825–2838.

[22] Andreas Geiger et al. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* (2013).

[23] Igor Gilitschenski et al. "Deep Orientation Uncertainty Learning based on a Bingham Loss". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=ryloogSKDS.

[24] Ross Girshick. "Fast R-CNN". In: *Proc. IEEE Int. Conf. Comp. Vis.* 2015, pp. 1440–1448.

[25] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2014, pp. 580–587.

[26] Lili Guo, Dan Xu, and Zhenping Qiang. "Background Subtraction Using Local SVD Binary Pattern". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2016, pp. 1159–1167. DOI: 10.1109/CVPRW.2016.148.

[27] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. "Multiple Choice Learning: Learning to Produce Multiple Structured Outputs." In: *NIPS*. Vol. 1. 2. Citeseer. 2012, p. 3.

[28] David Hall et al. "Probabilistic Object Detection: Definition and Evaluation". In: (Nov. 2018).

[29] David Hall et al. "Probabilistic Object Detection: Definition and Evaluation". In: Mar. 2020, pp. 1020–1029. DOI: 10.1109/WACV45572.2020.9093599.

[30] Ali Harakeh, Michael Smart, and Steven L. Waslander. "BayesOD: A Bayesian Approach for Uncertainty Estimation in Deep Object Detectors". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 87–93.

[31] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* June 2016.

[32] Kaiming He et al. "Mask R-CNN". In: *Proc. IEEE Int. Conf. Comp. Vis.* 2017, pp. 2961–2969.

[33] Yihui He et al. "Bounding box regression with uncertainty for accurate object detection". In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*. 2019, pp. 2888–2897.

[34] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=Sy2fzU9gl.

[35] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[36] Zhihang Hu et al. "A 3D Atrous Convolutional Long Short-Term Memory Network for Background Subtraction". In: *IEEE Access* 6 (2018), pp. 43450–43459. DOI: 10.1109/ACCESS.2018.2861223.

[37] Eric Jang et al. "Grasp2Vec: Learning Object Representations from Self-Supervised Grasping". In: *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 99–112.

[38] Won-Dong Jang et al. "Learning Vector Quantized Shape Code for Amodal Blastomere Instance Segmentation". In: *CoRR* abs/2012.00985 (2020). arXiv: 2012.00985. URL: https://arxiv.org/abs/2012.00985.

[39] Pakorn KaewTrakulPong and R. Bowden. "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection". In: 2002.

[40] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014. arXiv: http://arxiv.org/abs/1312.6114v10 [stat.ML].

[41] Alexander Kirillov et al. "Segment Anything". In: *arXiv:2304.02643* (2023).

[42] Simon A. A. Kohl et al. "A Probabilistic U-Net for Segmentation of Ambiguous Images". In: *CoRR* abs/1806.05034 (2018). arXiv: 1806.05034. URL: http://arxiv.org/abs/1806.05034.

[43] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *arXiv preprint 1612.01474* (2016).

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

[45] Ke Li and Jitendra Malik. "Amodal Instance Segmentation". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 677–693. ISBN: 978-3-319-46475-6.

[46] Xiang Li et al. "Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 21002–21012. URL: https://proceedings.neurips.cc/paper/2020/file/f0bda020d2470f2e74990a07a607ebd9-Paper.pdf.

[47] Chung-Ching Lin et al. "Video Instance Segmentation Tracking With a Modified VAE Architecture". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 13144–13154. DOI: 10.1109/CVPR42600.2020.01316.

[48] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* July 2017.

[49] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *Proc. Eur. Conf. Comp. Vis.* Springer. 2014.

[50] YuXuan Liu, Xi Chen, and Pieter Abbeel. *Self-Supervised Instance Segmentation by Grasping*. 2023. arXiv: 2305.06305 [cs.CV].

[51] YuXuan Liu et al. "Modeling Uncertainty and High Confidence Predictions with Latent-MaskRCNN". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023.

[52] Yuxuan Liu et al. "Autoregressive Uncertainty Modeling for 3D Bounding Box Prediction". In: *Computer Vision - ECCV 2022 - 17th European Conference*. 2022.

[53] Yuxuan Liu et al. "Distributional Instance Segmentation: Modeling Uncertainty and High Confidence Predictions with Latent-MaskRCNN". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023.

[54] Ze Liu et al. "Group-free 3d object detection via transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2949–2958.

[55] Ze Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.

[56] Martin Lohmann et al. "Learning about Objects by Learning to Interact with Them". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada, 2020. ISBN: 9781713829546.

[57] Jeffrey Mahler et al. "Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2017), pp. 1–8.

[58] Luke Metz et al. "Discrete sequential prediction of continuous actions for deep rl". In: *arXiv preprint arXiv:1705.05035* (2017).

[59] Gregory P. Meyer and Niranjan Thakurdesai. "Learning an Uncertainty-Aware Object Detector for Autonomous Driving". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), pp. 10521–10527.

[60] Gregory P. Meyer et al. "LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving." In: *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 12677–12686. URL: `http://dblp.uni-trier.de/db/conf/cvpr/cvpr2019.html`.

[61] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV*. 2020.

[62] Dimity Miller et al. "Benchmarking Sampling-based Probabilistic Object Detectors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.

[63] Ishan Misra, Rohit Girdhar, and Armand Joulin. "An End-to-End Transformer Model for 3D Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2906–2917.

[64] Arsalan Mousavian et al. "3d bounding box estimation using deep learning and geometry". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 7074–7082.

[65] Davy Neven et al. "Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[66] Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[67] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: `2203.02155 [cs.CL]`.

[68] Deepak Pathak et al. "Learning Instance Segmentation by Interaction". In: *CVPR Workshop on Benchmarks for Deep Learning in Robotic Vision*. 2018.

[69] Valentin Peretroukhin et al. "A Smooth Representation of SO(3) for Deep Rotation Learning with Uncertainty". In: *Proceedings of Robotics: Science and Systems (RSS'20)*. July 12–16, 2020.

[70] Charles R Qi et al. "Frustum pointnets for 3d object detection from rgb-d data". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 918–927.

[71] Charles R Qi et al. "Imvotenet: Boosting 3d object detection in point clouds with image votes". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[72] Charles R. Qi et al. "Deep Hough Voting for 3D Object Detection in Point Clouds." In: *ICCV*. IEEE, 2019, pp. 9276–9285. ISBN: 978-1-7281-4803-8. URL: `http://dblp.uni-trier.de/db/conf/iccv/iccv2019.html`.

[73] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2016, pp. 779–788.

[74] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Proc. Advances in Neural Inf. Process. Syst.* 2015, pp. 91–99.

[75] Robin Rombach et al. "High-Resolution Image Synthesis With Latent Diffusion Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 10684–10695.

[76] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. "FCAF3D: Fully Convolutional Anchor-Free 3D Object Detection". In: *arXiv preprint arXiv:2112.00322* (2021).

[77] Lorenz Rumberger, Lisa Mais, and Dagmar Kainmueller. "Probabilistic Deep Learning for Instance Segmentation". In: Jan. 2020, pp. 445–457. ISBN: 978-3-030-66414-5. DOI: `10.1007/978-3-030-66415-2_29`.

[78] Christian Rupprecht et al. "Learning in an uncertain world: Representing ambiguity through multiple hypotheses". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3591–3600.

[79] S Shi, X Wang, H PointRCNN Li, et al. "3d object proposal generation and detection from point cloud". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA*. 2019, pp. 16–20.

[80] Shaoshuai Shi et al. "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 10526–10535. DOI: 10.1109/CVPR42600.2020.01054. URL: https://openaccess.thecvf.com/content%5C_CVPR%5C_2020/html/Shi%5C_PV-RCNN%5C_Point-Voxel%5C_Feature%5C_Set%5C_Abstraction%5C_for%5C_3D%5C_Object%5C_Detection%5C_CVPR%5C_2020%5C_paper.html.

[81] Uriel Singer et al. "Text-To-4D Dynamic Scene Generation". In: *arXiv:2301.11280* (2023).

[82] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. "Sun rgb-d: A rgb-d scene understanding benchmark suite". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576.

[83] Ryutaro Tanno et al. "Learning From Noisy Labels by Regularized Estimation of Annotator Confusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[84] Zhi Tian et al. "Fcos: Fully convolutional one-stage object detection". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9627–9636.

[85] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133.

[86] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks". In: *International conference on machine learning*. PMLR. 2016, pp. 1747–1756.

[87] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[88] Zonghan Wu et al. *A Comprehensive Survey on Graph Neural Networks*. 2019. URL: http://arxiv.org/abs/1901.00596.

[89] Enze Xie et al. "PolarMask: Single Shot Instance Segmentation With Polar Representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[90] Qian Xie et al. "MLCVNet: Multi-Level Context VoteNet for 3D Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[91] Lin Yen-Chen et al. "NeRF-Supervision: Learning Dense Object Descriptors from Neural Radiance Fields". In: *IEEE Conference on Robotics and Automation (ICRA)*. 2022.

[92] Houjian Yu and Changhyun Choi. "Self-Supervised Interactive Object Segmentation Through a Singulation-and-Grasping Approach". In: *European Conference on Computer Vision*. 2022.

[93] Tianhe Yu et al. "Scaling Robot Learning with Semantically Imagined Experience". In: *arXiv preprint arXiv:2302.11550*. 2023.

[94] Andy Zeng et al. "Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2018.

[95] Cheng Zhang et al. "Learning with Free Object Segments for Long-Tailed Instance Segmentation". In: *Computer Vision - ECCV 2022 - 17th European Conference*. 2022.

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

[97] Yuanxin Zhong, Minghan Zhu, and Huei Peng. "Uncertainty-Aware Voxel based 3D Object Detection and Tracking with von-Mises Loss". In: *ArXiv* abs/2011.02553 (2020).

[98] Yi Zhou et al. "On the Continuity of Rotation Representations in Neural Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 5738–5746.

[99] Z. Zivkovic. "Improved adaptive Gaussian mixture model for background subtraction". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 2. 2004. DOI: 10.1109/ICPR.2004.1333992.

[100] Xueyan Zou et al. "Segment Everything Everywhere All at Once". In: *ArXiv* abs/2304.06718 (2023).

Appendix A

Appendix for Autoregressive 3D Bounding Box

A.1 Model Architecture and Training

Autoregressive 3D Bounding Box Estimation

For bounding box estimation, our model operates on 2D detection patch outputs of size 96 x 96. We take the 2D bounding box from object-detection to crop and resize the following features for each object: 3D point cloud, depth uncertainty score, normals, instance mask, amodal instance mask (which includes the occluded regions of the object). We normalize each point p in the point cloud with the 0.25 (Q_1) and 0.75 (Q_3) quantiles per dimension using $\frac{p-c_0}{s}$ for $c_0 = \frac{Q_1+Q_3}{2}$, $s = Q_3 - Q_1$. We omitted RGB since we found it wasn't necessary for training and improved generalization.

We stack each 2D feature along the channel dimension and embed the features using a 2D Resnet U-Net. The features from the top of the U-Net are used in a series of self-attention modules across embeddings from all objects in a scene so that information can be shared across objects. The resulting features from self-attention are tiled across the spatial dimension before the downward pass of the U-Net. Finally, the features from the highest spatial resolution of the U-Net are passed into several strided-convs, flattened, and projected to a 128-dimension feature h per object. Figure A.1 shows the overview of our model architecture.

For the autoregressive layers, we use 9 MLPs with hidden layers (128, 256, 512, 1024). For baselines, we keep the same architecture through h and use different sized MLPs depending on the box parameterization. We train using Adam with learning rate 1e-5 with a batch size of 24 scenes per step with varying number of objects per scene. We train for 10000 steps or until convergence.

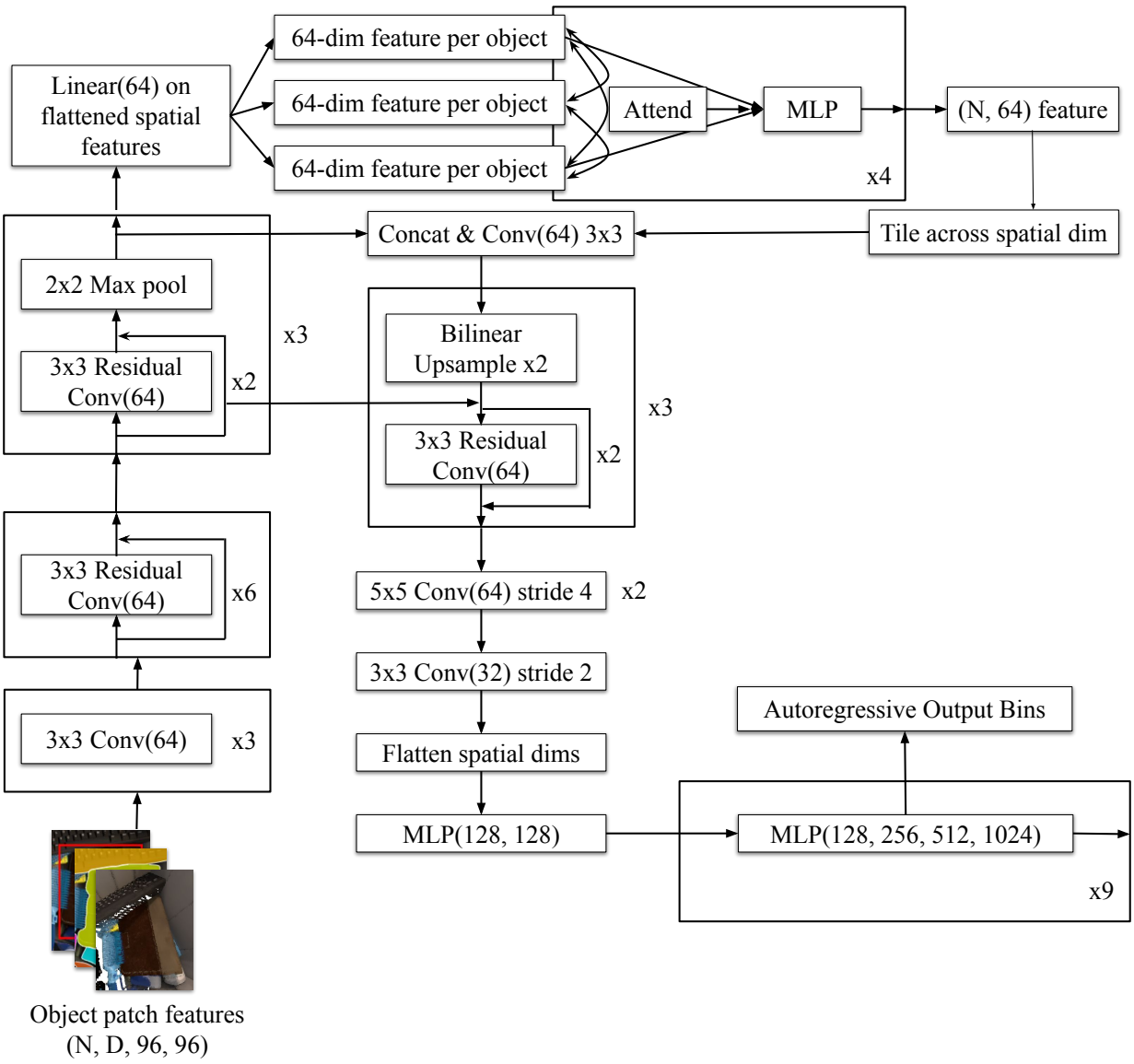

Figure A.1: Overview of Autoregressive Bounding Box Estimation architecture

Autoregressive 3D Object Detection

For Autoregressive FCAF3D, we add 7 autoregressive MLPs with hidden dimensions (128, 256, 512). All other parameters of FCAF3D are the same and we train the same hyper-parameters as the released code for 30 epochs. For the baseline FCAF3D, we trained the author-released model for 30 epochs on 8 gpus. We found that the benchmarked numbers for $AP_{0.25}$ and $AP_{0.50}$ were slightly lower than the reported ones in the original paper, so

in our table, we use the reported average AP across trials from the original paper. AP_{all} was calculated in a similar way as in MS-COCO by averaging AP for iou thresholds over $0.05, 0.10, 0.15, ..., 0.95$.

A.2 Quantile Box

Proof of Quantile-Confidence Box

Proof Sketch:

Let $P(b)$ be a distribution over an ordered set of boxes where for any two distinct boxes b_1, b_2 in the sample space, one must be contained in the other, $b_1 \subset b_2$ or $b_2 \subset b_1$. We'll show that a quantile box b_q is a confidence box with $p = 1 - q$ by 1) constructing a confidence box b_p for any given q, 2) showing that any $x \in b_p$ must have $O(x) > q$, and 3) therefore $b_p \subseteq Q(q) \subseteq b_q$ so the quantile box is a confidence box.

1) Confidence Box:

For any $p = 1-q$, we'll show how to construct a confidence box b_p. Using the ordered object distribution property of $P(b)$, we can define ordering as containment $b_1 < b_2 \equiv b_1 \subset b_2$. This ordering defines an inverse cdf:

$$F^{-1}(p) = \inf\{x : P(b \leq x) \geq p\} \tag{A.1}$$

Let $b_p = F^{-1}(1-q)$ be the inverse cdf of p; by definition b_p is a confidence box with confidence p since $P(b \leq b_p) = P(b \subseteq b_p) \geq p$

2) Occupancy of b_p:

We'll show that any $x \in b_p$ satisfies $O(x) > 1-p$. First we'll prove that that $P(b \geq b_p) > 1-p$. Let $b_0 = \inf\{b : b < b_p\}$, the smallest box that is strictly contained in b_p. (If no such b_0 exists, then b_p must be the smallest box in the distribution order such that $P(b \geq b_p) = 1$ and $P(b \geq b_p) > 1 - p$ for $p \neq 0$)

Since b_p is the inverse cdf of p, we know that $P(b \leq b_0) < p$, otherwise b_0 would be the inverse cdf of p (i.e. $b_0 = b_p$ a contradiction). It follows that

$$\begin{aligned} P(b \geq b_p) &= P(b > b_0) && \text{(A.2)} \\ &= 1 - P(b \leq b_0) && \text{(A.3)} \\ &> 1 - p && \text{(A.4)} \end{aligned}$$

Now consider any point $x \in b_p$:

$$O(x) = P(x \in b) \tag{A.5}$$
$$= \int_b \mathbb{1}\{x \in b\} p(b) db \tag{A.6}$$
$$\geq \int_{b \geq b_p} \mathbb{1}\{x \in b\} p(b) db \tag{A.7}$$
$$= \int_{b \geq b_p} p(b) db \tag{A.8}$$
$$= P(b \geq b_p) \tag{A.9}$$
$$> 1 - p \tag{A.10}$$

Where (A.7) follows from the nonegativity of $\mathbb{1}\{x \in b\} p(b)$. (A.8) follows from $x \in b_p$, $b_p \subseteq b$ which implies $x \in b$.

3) Quantile-Confidence Box:

Since any $x \in b_p$ satisfies $O(x) > 1 - p$, it follows that $b_p \subseteq Q(1-p)$, where $Q(q) = \{x : O(x) > q\}$ is the occupancy quantile with quantile q. The quantile box by construction must contain the occupancy quantile $Q(q) \subseteq b_q$, therefore we have $b_p \subseteq Q(1-p) \subseteq b_q$, and

$$P(b \subseteq b_q) \geq P(b \subseteq b_p) \tag{A.11}$$
$$\geq p \tag{A.12}$$

So b_q is a confidence box with confidence requirement p.

Quantile Box Algorithm

We propose a fast quantile box Algorithm 1 that runs in polynomial time and is easily batchable on GPU. We use a finite sample of k boxes to approximate the occupancy and a sample of km points to approximate the occupancy quantile $Q(q)$. To find the minimum volume box, we assume that one of the sampled box rotations will be close to the optimal quantile box rotation. We take the sampled rotations and calculate the rotation-axis-aligned bounding box volume for the occupancy quantile. The minimum volume rotation is selected for the quantile box and corresponding dimension/center calculated accordingly.

Empirically we find that $k = 64$, $m = 4^3$ provides a good trade-off of variance and inference time. We can efficiently batch all operations on GPU, and find that quantile box inference for 15 objects takes no more than 10ms on a NVIDIA 1080TI.

A.3 Dataset

Our dataset consists of almost 7000 simulated scenes of common objects in bins. See Figure A.2 for examples. Each scene consists of the following data:

Algorithm 1: Quantile Box Algorithm

Given: quantile q, box distribution $P(b|h)$, numbers of box samples k, number of point samples m

Sample $b^{(1)}, ..., b^{(k)} \sim P(b|h)$ boxes

For each $b^{(i)}$, sample m random points within $b^{(i)}$, adding all points to a set T

For all $x \in T$, estimate $O(x) = \frac{1}{k} \sum_i^k \mathbb{1}\{x \in b^{(i)}\}$

Construct the occupancy quantile $Q(q) = \{x \in T : O(x) > q\}$

for $b^{(1)}, ..., b^{(k)}$ **do**

  Let R_i be the rotation of $b^{(i)}$

  Compute the volume of the $Q(q)$ bounding box under R_i,

  $v_i = \prod_{a \in x,y,z} (\max_{x \in Q(q)} (R_i^{-1} x)_a - \min_{x \in Q(q)} (R_i^{-1} x)_a)$

Find the minimum volume box $i^* = \arg\min_i v_i$

Let $s_a = \max_{x \in Q(q)} (R_{i^*}^{-1} x)_a$, $t_a = \min_{x \in Q(q)} (R_{i^*}^{-1} x)_a$

Return box $b = (d, c, R_{i^*})$ with dimensions $d = (t_x - s_x, t_y - s_y, t_z - s_z)$ and center $c = R_{i^*}(s_x + d_x/2, s_y + d_y/2, s_z + d_z/2)$

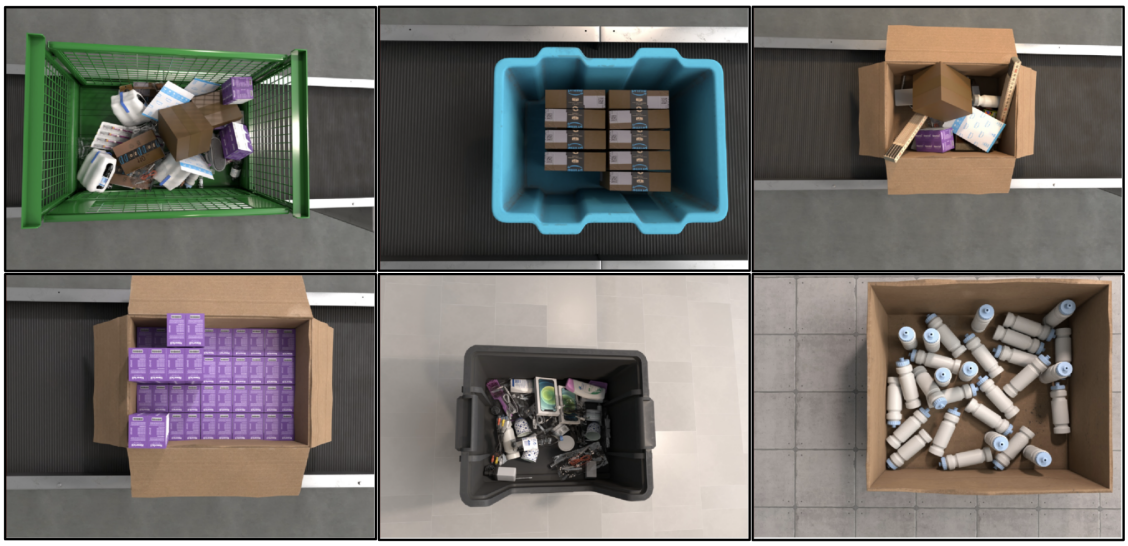

Figure A.2: Examples of scenes from our dataset

- **RGB** image of shape (H, W, 3)
- **Depth** map of shape (H, W)
- **Intrinsic Matrix** of the camera (3, 3)
- **Normals Map** of shape (H, W, 3)
- **Instance Masks** of shape (N, H, W) where N is the number of objects

- **Amodal Instance masks** of shape (N, H, W) which includes the occluded regions of the object
- **3D Bounding Box** of each object (N, 9) as determined by dimensions, center, and rotation.

A.4 Visualizations

In this section, we show various qualitative comparisons and visualization of our method.

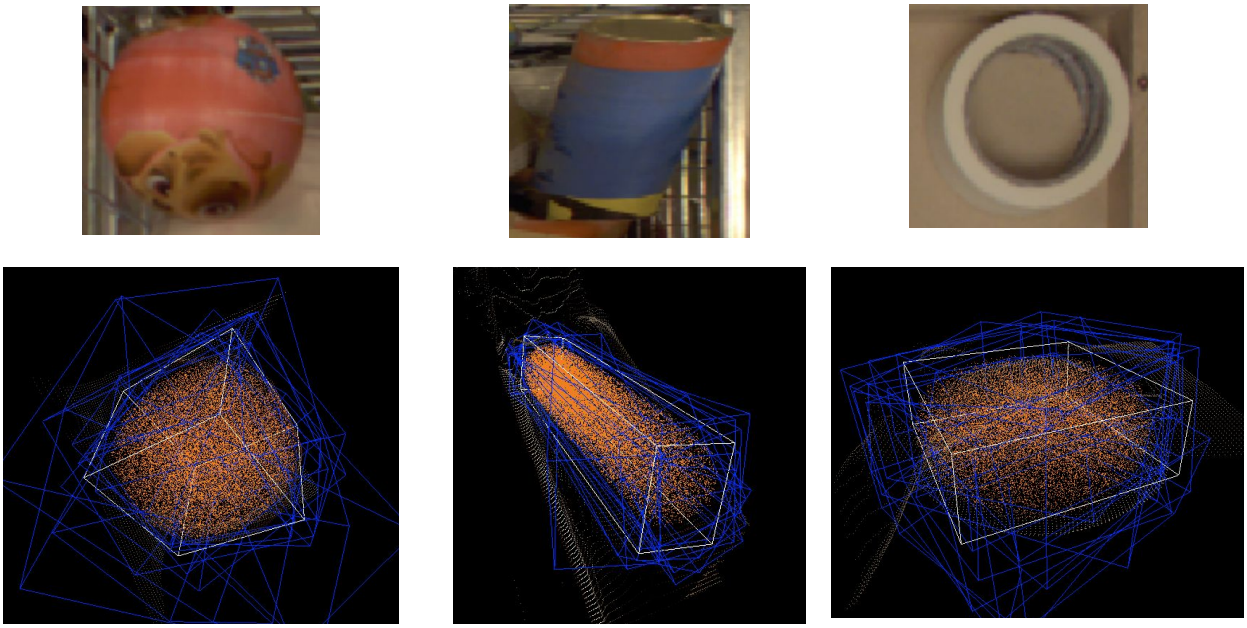

Figure A.3: Visualization of our model predictions on objects with rotational symmetry. The blue boxes show various samples from our model. The orange point cloud is the occupancy quantile. The white box is the quantile box.

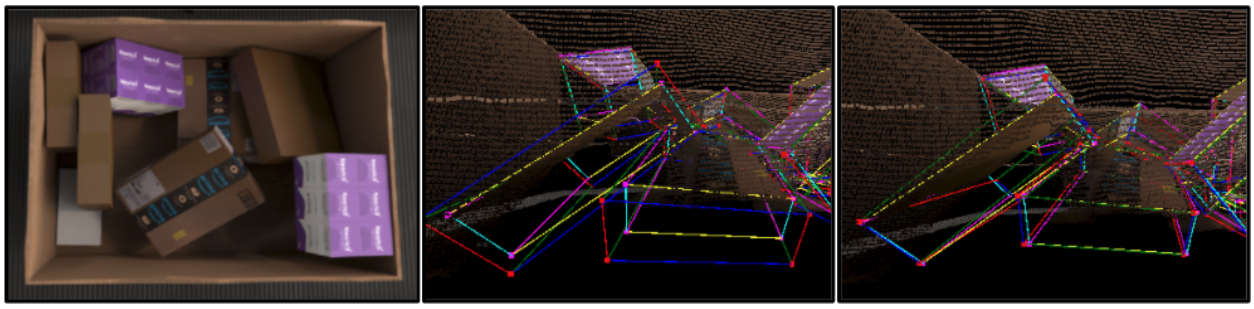

Figure A.4: Visualization of our dimension conditioning method. The model is able to leverage the conditioning information to accurately predict the correct pose & dimension for each object's 3D bounding box. The prediction is shown in red-blue-green and the ground truth in turquoise-yellow-pink. Left: image of the scene. Middle: vanilla beam search. Right: beam search with dimension conditioning.

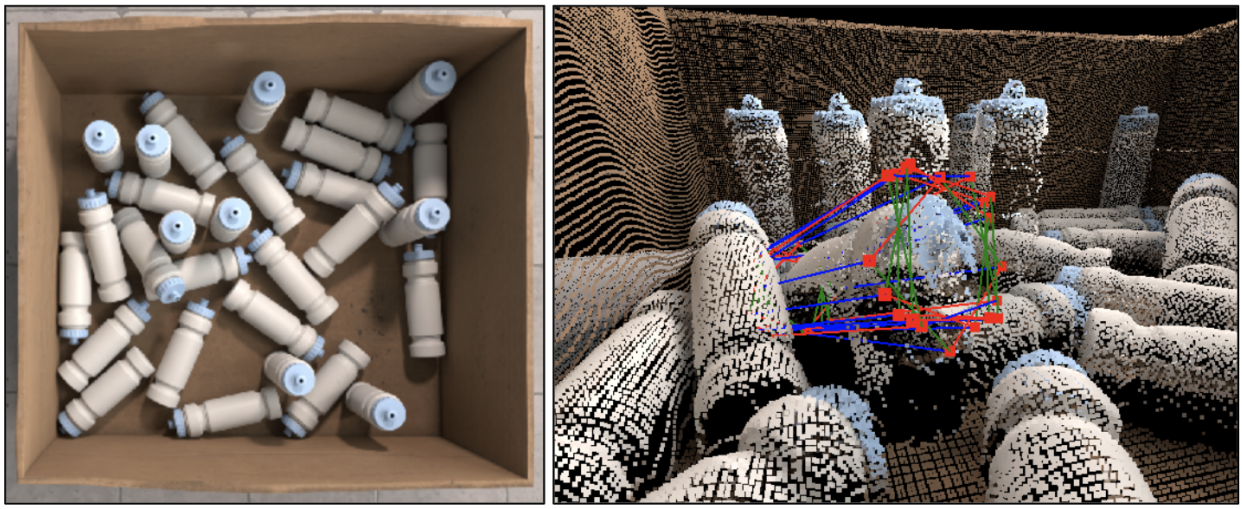

Figure A.5: Visualization of bounding box samples from our autoregressive model on a rotationally symmetric water bottle. Our model is able to sample different modes for symmetric objects whereas a deterministic model would only be able to predict a single mode.

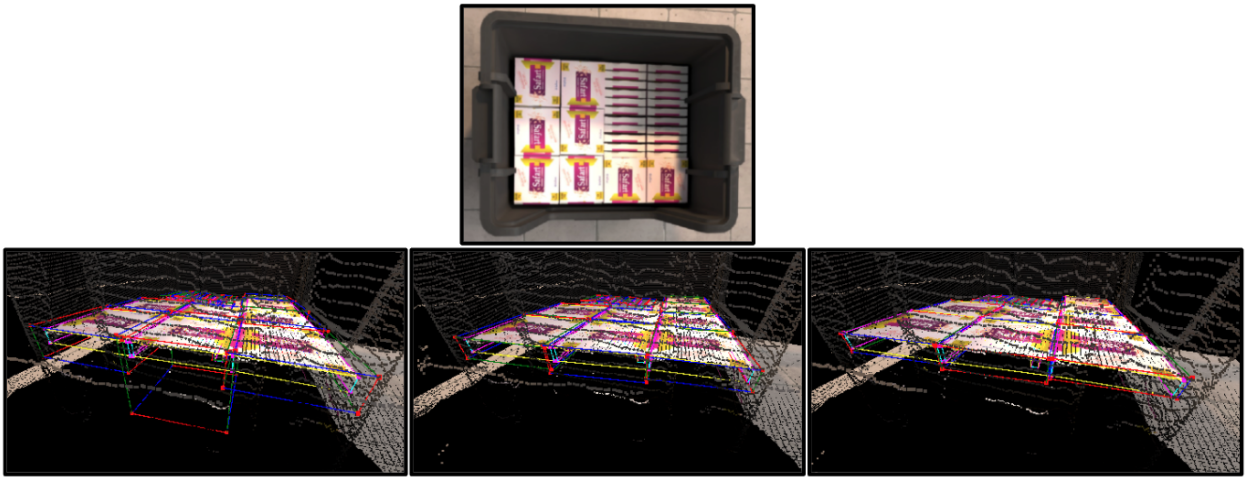

Figure A.6: Visualization of bounding box predictions with different quantiles. We can see that lower quantiles lead to larger boxes in the direction of uncertainty. Top: image of the scene. Left: quantile 0.1 Middle: quantile 0.3. Right: quantile 0.5.

Appendix B

Appendix for Latent-MaskRCNN

B.1 Latent-MaskRCNN Architecture and Training

In this section, we explain the architecture of Latent-MaskRCNN (Section 3.4) in more detail. See Figure B.1 for a visual overview.

In our experiments, we use latent codes of a fixed vector dimensionality $z \in \mathbb{R}^d$. We found that $d = 64$ was a reasonable choice that worked for all datasets.

In the decoder (Figure B.1 (a)), we tile the latent codes across the spatial dimensions of the image and concatenate them with the feature maps coming out of the FPN. Then we use a few convolutional layers to project the combined feature maps back down to their original channel dimensionality. The rest of the model (region-proposal, classifier head, mask head) uses these latent-augmented feature maps but is otherwise unchanged from MaskRCNN. As we discussed in Section 3.4, this scheme allows the latent codes to influence every stage of the prediction.

The encoder (Figure B.1 (b)) produces a Gaussian distribution over latent codes based on both the image and the ground-truth instances. For each instance y_i, we extract RoI-aligned features from the FPN feature maps, as well as of the ground-truth instance masks. Then we use a small CNN to embed each one into a single feature vector. At this point, we want to accumulate information from among the per-instance features, and produce a latent representation of a fixed size. To do this, we employ a graph neural network (GNN, [88]), where each instance constitutes a node in a fully-connected graph. After several graph network layers, we mean-pool across the node features and use a fully-connected layer to produce a mean and log-variance for our latent code.

The prior (Figure B.1 (c)) also predicts a Gaussian distribution over latent code, but it does not have access to the ground-truth instances. Instead, we apply a few convolutional layers to the FPN feature maps, mean-pool across the spatial dimensions, and then predict a mean and log-variance using a small MLP.

Consider the following training objective, and observe that it is equivalent to the standard

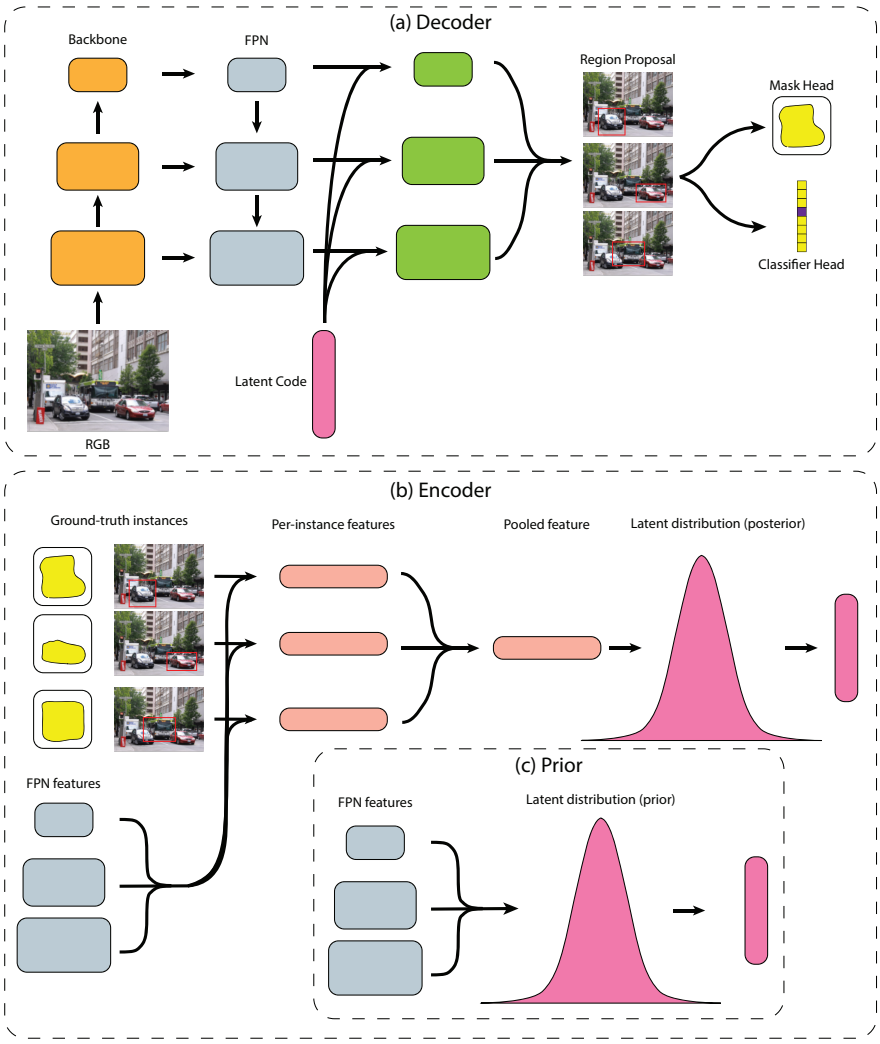

Figure B.1: The architecture of Latent-MaskRCNN. (a) The decoder is exactly the same as MaskRCNN, except that, before region proposal, we augment the FPN feature maps with a latent code. (b) The encoder takes the (non-augmented) FPN feature maps and a list of ground-truth instances and predicts a diagonal Gaussian distribution over latent codes. (c) The prior takes the (non-augmented) FPN feature maps and predicts a diagonal Gaussian distribution over latent codes.

VAE objective when $\beta = 1$.

$$\mathcal{L}(x, y) = \mathbb{E}_{z \sim q(z|y,x)}[\log p(y|x, z)] - \beta D_{KL}(q(z|y, x)||p(z|x))$$

When training VAEs, it is common to use a KL warmup, where β is ramped up from 0 over the first few epochs of training. This helps encourage the latent codes to become more

informative since there is no penalty for using them at the beginning of training (when the encoder is untrained and they are uninformative).

We trained all models on a single 8-GPU machine (1080Ti) of our internal cluster. Training takes roughly 20-30 hours for each model, depending on the dataset (on par with MaskR-CNN). Our code is available at `https://segm.yuxuanliu.com/`

B.2 Datasets

We are releasing the Apparel-5k dataset at `https://segm.yuxuanliu.com/`. This data was collected over the course of robot picking items between bins. No personally identifiable information or human data was collected. We also use COCO and Cityscapes which are licensed for academic use.

B.3 Metrics

In this section, we describe in detail the application-specific metrics that we used in Section 3.5.

First, we provide a brief overview of the mAP (mean average precision) metric that is commonly used to evaluate instance segmentation predictions. We say that a predicted instance and a ground-truth constitute a match if they have the same class, and their masks are within some IoU threshold of each other. We iterate through the predicted instances in order of decreasing confidence and determine which ones have a matching ground-truth instance (note that each ground-truth instance can only appear in one match). The predicted instances that appear in a match are considered true positives, and the remaining ones are considered false positives. We can then plot a precision-recall curve using this information, and compute the area under the curve. Typically this is done for each class, and we obtain the *average precision* (AP) by averaging across classes. Note that the AP still depends on the IoU threshold that we choose. The *mean average precision* (mAP) that is typically reported is the AP averaged across several IoU thresholds $(0.5, 0.55, \ldots, 0.9, 0.95)$.

mAP is a well-balanced metric, in the sense that it equally penalizes over-segmentation and under-segmentation, and considers both precision and recall. However, for specific applications like the ones we discussed in Sections 3.6 and 3.5, we may want different metrics that better reflect the asymmetric costs of different types of errors.

For the high-precision use cases like the one discussed in Section 3.6, we considered the *max recall at high precision* (MR@HP). We perform the matching procedure in a similar manner to mAP, except that we use IoP instead of IoU. Next, for a given precision threshold p_i and IoP threshold τ_j, we can compute the recall that each model achieves (or zero, if it never achieves precision p_i). The MR@HP metric is the average of these recalls, over a range

Algorithm 2: Confidence Mask

Given: confidence requirement p, k samples of y each with masks $m_j \in y$
Initialize $M \leftarrow \{\}$ to be a set of predicted masks
while *the masks predicted in* M *exceed some score* **do**
  for $m_h \in y_i$ **do**
    Compute the area of the intersection $I_{jg} = |m_h \cap m_j \setminus M|$ with all other masks $m_j \in y_g$
    Let C be the set of masks with the kp highest I_{jg} where each mask must come from a unique y_g
    Compute the intersection $m_h^* = \bigcap_{m \in C} m \setminus M$ ignoring predicted masks M
  Add the mask with the highest score $\arg\max s_{m_h^*}$ to M

of precision threshold p and IoP thresholds τ:

$$\text{MR@HP} = \frac{1}{|p| \cdot |\tau|} \sum_{p_i \in p, \tau_j \in \tau} \max_{t:\text{Precision}(\tau_j) \geq p_i} \text{Recall}(t, \tau_j)$$

For the evaluation in Section 3.6, we use $p = \tau = [0.75, 0.8, 0.85, 0.9, 0.95]$.

AR was introduced in [49] and is gaining popularity in the instance segmentation community as a complement to mAP. It also uses the matching procedure that mAP does, but then it simply averages each method's recall over the standard range of IoU thresholds.

For high-recall use cases like the one discussed in Section 3.5, we considered the average recall (AR), but using IoG instead of IoU. This measures both recall (did the model predict all the instances?) while also penalizing over-segmentation (are there any predicted masks that are too small).

B.4 Algorithms

In Section 3.5, we proposed Confidence Masks and Union NMS as two applications of Latent-MaskRCNN. Algorithm 2 details the iterative greedy confidence mask prediction, and Algorithm 3 details the Union NMS procedure.

B.5 Calibrated Uncertainty

In Section 3.5, we proposed a scoring method for confidence mask predictions. Existing models such as MaskRCNN tend to be extremely confident in general, even when they are wrong. However, a distributionally-expressive model should be able to express more calibrated confidence scores.

To evaluate whether these scores correlate well with mask quality, we compare the computed score with the IoU between the predicted mask and the closest ground-truth mask

Algorithm 3: Union-NMS

Given: Masks $M = \{m_i\}_{i=1}^{n}$ sorted by confidence, IoU threshold τ
Initialize $S \leftarrow \{\}$ to an empty map
for $i = 1 : n$ **do**
  if $i \in K$ **then**
    continue
  $S[i] \leftarrow \{\}$
  for $j = i + 1 : n$ **do**
    if $IoU(m_i, m_j) > \tau$ **then**
      Add j to $S[i]$
Initialize $U \leftarrow \{\}$
for $i \in keys(S)$ **do**
  $m \leftarrow m_i$
  for $j \in S[i]$ **do**
    $m \leftarrow m \cup m_j$
  Add m to U
Result: U

Table B.1: ROC AUC using scores to predict ground truth IoU > 0.5

Method	COCO	Cityscapes	Apparel-5k
MaskRCNN	0.7372	0.8443	0.7756
Confidence Mask	**0.7941**	**0.8666**	**0.9435**

(which indicates how correct the mask is). Figure B.2 plots these two quantities on the Apparel-5k dataset. We see that MaskRCNN almost always has full confidence (regardless of the IoU), while the Latent-MaskRCNN produces confidence scores that are reasonably correlated with the mask accuracy.

Table B.1 also shows ROC AUC where we use each model's score to predict when the ground truth IoU exceeds some threshold $\text{IoU}(c_p, G) \geq 0.5$. Across all three datasets, Latent-MaskRCNN has higher AUCs compared to MaskRCNN, suggesting its scores are more predictive of mask quality. This suggests that a distributionally expressive model like Latent-MaskRCNN with the proposed confidence mask scoring can be more effective at measuring uncertainty.

When we proposed p-confidence masks in Section 3.6, we wanted each confidence mask to be contained within some object with probability p. To evaluate this, we can compare the fraction of predictions made by each confidence mask that have IoP > 0.95 with the confidence requirement p. Figure B.2 shows the IoP quantiles for different confidence mask p's on Apparel-5k. We generally find that increasing the confidence requirement, p, also increases the IoP of the predictions. Meanwhile, MaskRCNN can only achieve one distribution

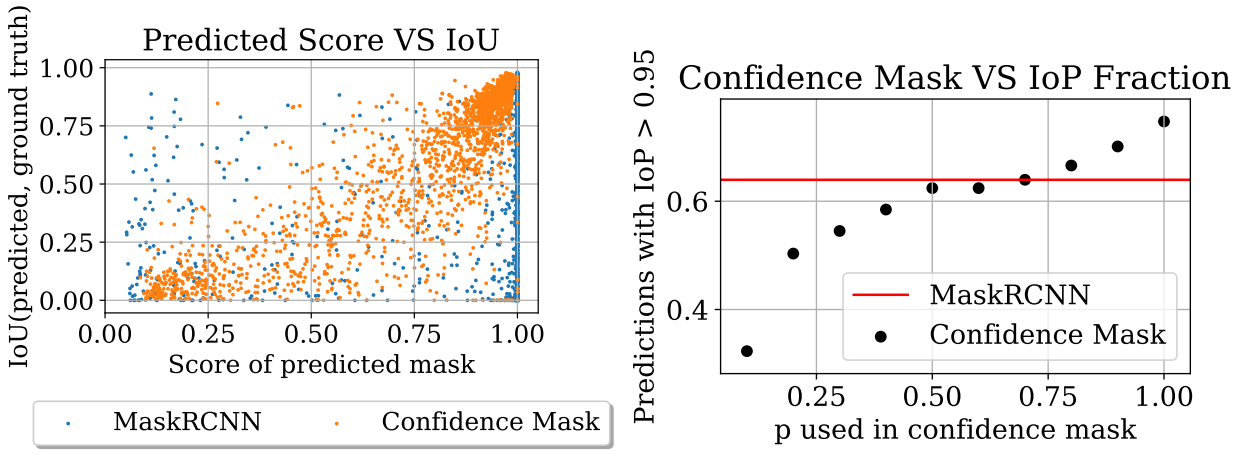

Figure B.2: Left: Correlation of predicted score with ground truth IoU of each mask on the Apparel-5k dataset. Latent confidence mask scores tend to be more correlated while MaskRCNN is overconfident. Right: Increasing the confidence requirement p increases the IoP of the predictions while MaskRCNN can only realize one IoP distribution.

of IoP's since no knobs control the confidence of its outputs.