Supplemental Materials for *Knolling Bot: Learning Robotic Object Arrangement from Tidy Demonstrations* IROS 2024 Submission

Yuhang Hu* Zhizhuo Zhang Xinyue Zhu Ruibo Liu Philippe Wyder Hod Lipson
Columbia University

March 13, 2024

1 Position Encodering

Our model utilizes two position encoding methods to process the input data effectively. The first method, aligned with the original transformer paper's approach, assigns unique tokens to each position in the list. This gives the data a sense of order, enabling the model to predict the target position of the initial objects and apply autoregression for the remaining objects. The second position encoding method maps the input data into a higher-dimensional space via sinusoidal functions. This method allows the model to handle higher frequency representations that potentially improve performance. The input, originally a 2-dimensional vector representing object length and width, is transformed into a 21-dimensional feature vector through this encoding method. By varying the wavelength at five different frequencies, the position encoding captures fine-grained patterns in the data, leading to more accurate predictions.

2 Two-Stage Self-Supervised Training

Our self-supervised learning framework for training knolling models follows a curriculum-based approach, progressing from simple to complex tasks. This two-stage training process is designed to first acquaint the model with the basics of object arrangement, and then refine its proficiency in executing complete knolling tasks from scratch.

2.1 Pre-Training Stage

In the pre-training phase, the model is exposed to partially organized scenes, where it needs to predict the placement of the next object given the current arrangement. Formally, let $O = o_1, o_2, \ldots, o_N$ be the set of N objects, and $P = p_1, p_2, \ldots, p_{N-1}$ be the given positions of the first $N-1$ objects. The pre-training objective is to predict the position p_N of the remaining object o_N:

$$p_N = f_{\text{pre-train}}(O, P) \tag{1}$$

This pre-training stage allows the model to learn the fundamental principles of spatial organization and object placement without the added complexity of predicting all object positions simultaneously.

2.2 Fine-Tuning Stage

The fine-tuning phase is dedicated to refining the model's ability to execute complete knolling tasks from scratch, starting with cluttered or disorganized scenes. In this stage, the model is tasked with predicting the positions $P = p_1, p_2, \ldots, p_N$ of all N objects given only their dimensions $O = o_1, o_2, \ldots, o_N$:

$$P = f_{\text{fine-tune}}(O) \tag{2}$$

By addressing the challenges of error accumulation and the differentiation of target positions for each object, this phase fine-tunes the model's precision and its ability to generate coherent, tidy arrangements from cluttered starting points.

2.3 Rationale

The two-stage training process is motivated by the principles of curriculum learning, where the model is gradually exposed to increasingly complex tasks, allowing it to build a solid foundation before tackling the full problem. By first learning the basics of object placement in the pre-training stage, the model can better generalize and refine its understanding during the fine-tuning phase, leading to improved performance and robustness in generating complete knolling solutions.

Additionally, this approach helps mitigate the compounding effects of error accumulation, which can be particularly detrimental in autoregressive prediction tasks like knolling. By first mastering the simpler task of predicting the next object placement, the model is better prepared to handle the complexities of predicting all object positions simultaneously during fine-tuning.

2.4 Ablation Experiment on Pre-training Effectiveness

To delve deeper into the efficacy of the pre-training process and its impact on the performance of our knolling model, we carried out an ablation study. This experiment was designed to discern the value added by the pre-training step in our model's pipeline.

We used 1M small rectangular-shaped object data for this experiment to ensure the results were derived from a comprehensive data pool. Two distinct training methodologies were adopted.

Table 1: Test Error vs Dataset Size

	Mean	**Std**	**Min**	**Max**
125K	6.46E-04	1.82E-03	1.39E-15	3.17E-02
250K	5.81E-04	1.81E-03	0.00E+00	3.21E-02
500K	5.18E-04	1.79E-03	1.39E-15	3.53E-02
1M	4.29E-04	1.61E-03	0.00E+00	3.49E-02
1M-Fine	3.57E-04	1.54E-03	0.00E+00	3.44E-02

Direct Training: The model was trained directly on the target task without any pre-training. Pre-training + Fine-tuning: In this approach, the model was first pre-trained on an auxiliary task and subsequently fine-tuned on the target knolling task, as described in the methodology section. The performance metrics obtained from the two training methods are presented in Table 1. The model that used the pre-training and fine-tuning process outperformed the model trained without pre-training, demonstrating the effectiveness of our curriculum-based training approach in improving the model's knolling capabilities.

3 Transformer Performance across Dataset Sizes

This experiment aimed to evaluate the performance of our knolling model across different scales of data. Four dataset sizes were selected for this study: 125k, 250k, 500k, and 1M. Each variant of the knolling model was trained independently until convergence. Following training, all models were evaluated using a consistent test dataset to maintain uniformity in the assessment.

The test errors obtained for each dataset size are shown in Table 1. The results highlight a consistent decline in error as the dataset size augments. This trend validates the widely accepted belief that transformers excel with the increase in data volume.

4 Supplementary Material: Visual Perception Model Evaluation

4.1 YOLO Segmentation Model

Our visual perception module employs a customized YOLO v8 model, fine-tuned for object detection, segmentation, and attribute prediction. We trained and evaluated two versions of the model: one for the simulated environment (YOLO-sim), and another for real-world deployment, (YOLO-real).

4.1.1 Training Settings

Simulation Model (YOLO-sim):

- Number of categories: 9
- Number of objects per image: 4-11
- Dataset sizes: 3200 train images, 800 test images
- Pre-trained model: yolov8n-seg

Real-World Model (YOLO-real):

- Number of categories: 9
- Number of objects per image: 4-11
- Dataset sizes: 80 train images, 20 test images
- Pre-trained model: YOLO-sim

4.1.2 Evaluation Metrics and Results

We evaluated the models' performance using the following metrics:

- Average Accuracy: Measures the accuracy of category predictions.
- Confusion Matrix: Provides a detailed breakdown of the model's category prediction performance.
- Average Hamming Distance: Assesses the quality of the predicted segmentation masks compared to the ground truth.

Table 2: YOLO Segmentation Model Evaluation Results

Metric	**YOLO-sim**	**YOLO-real**
Average Hamming Distance	0.06743	0.05831
Average Accuracy	0.999	0.9857

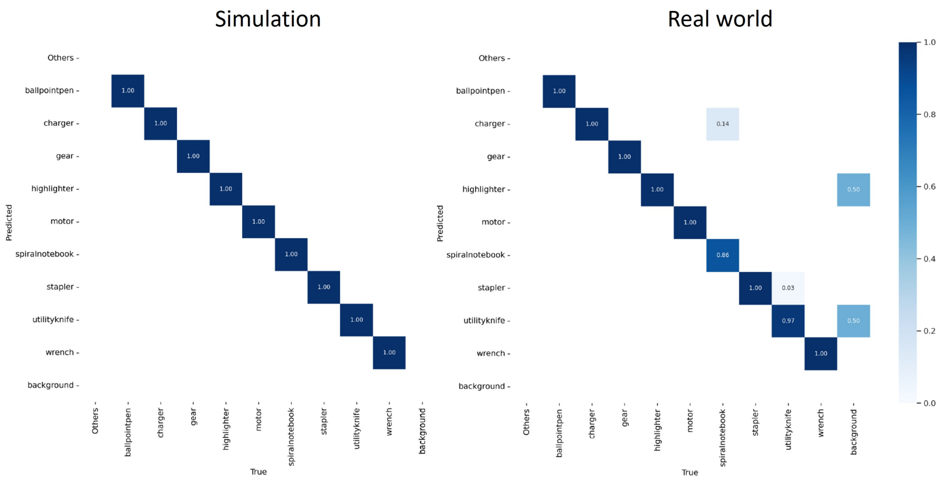

Figure 1: Confusion matrix for the YOLO segmentation model. The matrix shows the model's category prediction performance, with each cell representing the number of instances predicted as a particular category (column) compared to the true category (row). The diagonal cells indicate correct predictions, while off-diagonal cells represent misclassifications.

The evaluation results for both models are summarized in Tab.2.

The results demonstrate that both models achieve high accuracy in category prediction and generate precise segmentation masks, with the real-world model slightly outperforming the simulation model in terms of Average Hamming Distance. The confusion matrices for both models are provided in Fig.1.

These evaluation results validate the effectiveness of our visual perception module in accurately detecting, segmenting, and categorizing objects in both simulated and real-world environments, enabling robust input for the subsequent knolling pipeline.