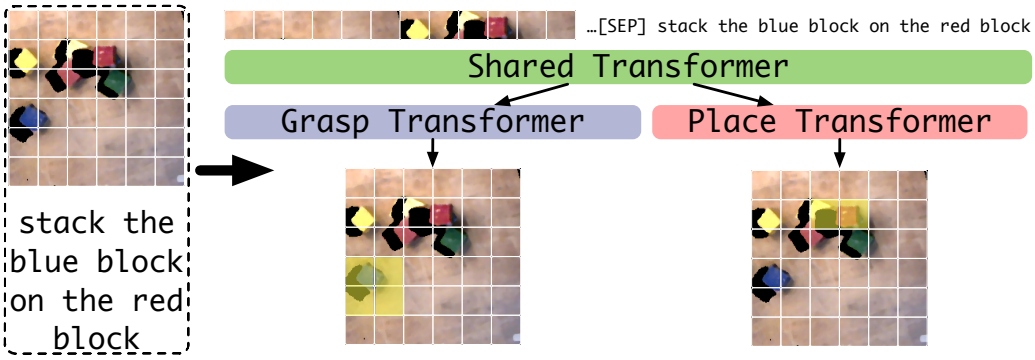


Figure 4: Our Transformer-based Visual Goal Prediction model, as run on real robot data. It takes an image and an instruction as input, then produces separate masks for grasping and placing.

A Model

A.1 Transformer

Figure 4 shows a schematic of our Transformer-based VGP model, run on real robot data. The inputs are a bird's eye color image (and depth image) and a language command. The image is tiled into an array of non-overlapping square patches, which are flattened into a one-dimensional list by concatenating each row of tiles. The tokens of the language command are then concatenated to the end of that list. This concatenated input is passed through a shared set of Transformer encoder layers, which encodes the dependencies between words and the image. The output of the shared Transformer is then passed to two identical Transformer encoders with separate parameters, one for predicting a grasp mask, and one for predicting a place mask. This bifurcation is based on the observation that there are commonalities between producing the two types of masks (e.g. localizing colors, delineating blocks) but that the final tasks of producing grasp and place masks are distinct.

A.2 Hyperparameters

The number of shared and combined layers is treated as a hyperparameter $n \in [0, 2, 4, 6]$, as well as the number of warmup steps $w \in [100, 400, 1000, 4000]$, the dropout probability $p \in [0.25, 0.33, 0.40]$, and the loss weight ratio $\gamma \in [0.01, 0.1, 0.2]$. Due to the larger number of hyperparameters, we sample 70 random parameter configurations.

A.3 Q-Learning

Our model relies on a learned function from states and actions to expected rewards, $Q(s_t, a)$. This function is learned in the context of a Markov Decision Process (S, A, P, R, γ) composed of a set of states S, a set of actions A, a transition function $P : S \times S \times A \to \mathbb{R}$, a reward function $R : S \times A \to \mathbb{R}$, and a discount factor γ, where $0 \leq \gamma \leq 1$. At time t, an agent observes state S_t and chooses action $a_t = \pi(s_t)$, where π is a policy. The function Q, approximating the expected reward for each state-action pair, can be used to extract a deterministic policy π that maximizes the expected reward: $\pi(s_t) = \operatorname{argmax} Q(s_t, a), \forall a \in A$. We learn Q to maximize the reward R over time by minimizing $|Q(s_t, a_t) - y_t|$, where $y_t = R(s_{t+1}, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}))$. In a multi-step task, the definition of R has a major impact on the learning efficiency of the agent. Specifically, relying on a single reward at the end of a task yields a very sparse reward signal and results in inefficient learning. In the multi-step tasks considered here (stacking and row-making) there are intermediate subtasks that naturally lend themselves to reward shaping, i.e. providing smaller intermediate rewards to the agent when the chosen action reflects an incremental progression towards the goal state. The SPOT-Q algorithm, introduced by Hundt et al. [9], uses intermediate rewards to shape the learning of $Q(s_t, a)$ for row-making and stacking. We use SPOT-Q to learn the function $Q(s_t, a)$. Given a pixel-wise $n \times n$ state space of an image and two actions (grasp and place), this function can be expressed as a tensor $Q \in \mathbb{R}^{n \times n \times 2}$.

B Experiment 1

B.1 Combination Heuristics

In the case of stacking, following Hundt et al. [9], we intersect each mask with a common-sense mask that assigns a value of 0 where there are no blocks (i.e. an empty space cannot be the location of a grasp or a place). For row-making, we allow placing in empty locations, but intersect the grasp mask with a common-sense mask. For place and grasp actions in stacking, and grasp actions in row-making, we then find the maximum location s^* in mask M: $s^* = \text{argmax}\, M$. Because the reconstruction function copies a patch value across all of its pixels, for a patch size of p there will be p^2 pixels with the value of s^*. We intersect this patch with the common-sense mask to find the single block with the highest value in M. For place actions in row-making, we simply threshold M to obtain a set of valid patches to intersect with the Q values. Both processes produce binary masks.

B.2 Simulation

We use the CopelliaSim simulator [49] for experiments involving a simulated robot. The simulated agent collects observations via a fixed RGB-D camera, whose images are projected to a birds-eye view, as shown in Hundt et al. [9]. The agent operates over a discretized spatial and angular action space, and movement to a particular location and angle is performed by an inverse kinematics solver built into the simulator.

Images All real images were collected via a UR-5 robot, Robotiq 2f-85 2-finger gripper, and Prime-sense Carmine RGB-D camera. All of these conditions are different from the simulator, except the UR5. The images were collected under varied lighting conditions. The data is saved in the same file format as the experiments in Tables 1 and 2. with a birds-eye view, 224×224 RBG and depth-map images, etc.

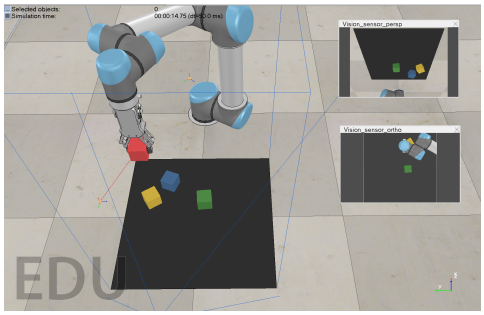


Figure 5: Simulation Environment

Early Termination The step-by-step nature of the instructions means that the agent is sometimes unable to reverse course: for example, if the first instruction were "stack the green block on the blue block" but the agent mistakenly grasped the red block and placed it on the green block, the green block would become unreachable. Since neither the Q-value model nor the language understanding model were trained on unstacking tasks, there would be no way to reverse course here without adding an external observer. Thus, in these cases, we terminate the trial. Similarly, if the stack is toppled, we terminate the trial, as without an external observer we cannot determine which step of the process we have regressed to, since the toppling of partial stacks is not unusual. Since blocks are typically not occluded in row-making, we do not need these heuristics, and the only way for a trial to fail is by timing out after 30 actions.

Simulator For experiments involving a simulated robot, the CopelliaSim simulator [49] was used. The simulated agent collects observations via a fixed RGB-D camera, whose images are project to a birds-eye view. The agent operates over a discretized spatial and angular action space, and movement to a particular location and angle is performed by an inverse kinematics solver built into the simulator.

B.3 Error analysis

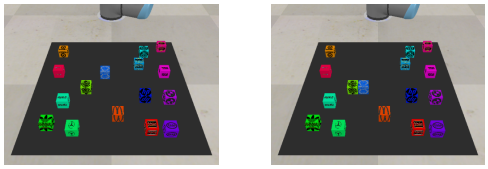
As mentioned in §5.1, stacking tasks may terminate early for a number of reasons, including irreversible actions that make a successful stack impossible. We conduct an error analysis to determine what percentage of the failures are due to errors in the perception/reasoning component (i.e. the masking) and what percentage are due to physical failures of the grasping arm. We find that 61.97% of the failures could be attributed to timing out after 30 actions. Qualitatively, we found that this

Dataset	Configs	Types	Tokens	Utterances	Mean Length
Bisk et al. [10]	100	1,820	233,544	12,975	18.0
Physically Feasible	100	1,434	133,083	7,398	18.0

Table 6: Dataset Statistics for feasible subset of the Bisk et al. [10] data

"There is a disconnected "square" at the bottom. Place Shell on top of the lower left corner."

Figure 6: A particularly challenging example for the model. Not only does this involve a higher-order concept (disconnected square) but it also incorporates noun-phrase ellipsis.



"Add Shell as the second block in the four-block row."

Figure 7: Ambiguous command makes reference to future steps: a row of 4 is later constructed, but at the time of construction, the agent has no access to this information.

often happens when a block tumbles outside of the work area, rendering it impossible to grasp. Another 7.04% can be attributed to toppled stacks, where a stack height of 2 or 3 was reached before a place action destroyed the whole stack, ending the trial. Finally, 30.99% were due to incorrect block orderings which occluded crucial blocks; this is due to the 21.42% of grasp actions that picked up the incorrect block.

B.4 Proof-of-Concept

Due to restrictions induced by the ongoing COVID-19 crisis, we were unable to run sufficient trials to include quantitative real results. Nevertheless, we include a proof-of-concept demonstration video in which we run the Transformer-based masking model with a real robot arm to successfully complete a real stack in the bottom to top order green, yellow, blue, red.

C Experiment 2

C.1 Data Translation and Preprocessing

Note that in the original dataset, 4.82% of the examples have multiple blocks moving in a single frame. As this breaks the assumptions made in § 1, we ignore these examples both while training and evaluating.

For training, we convert the 64×64 state image grids into binary masks, where all elements are zero except the pixels corresponding to the block which is moved.

C.2 Data Filtering

We filter out physically infeasible examples, which are typically due to unstable structures toppling or blocks being placed in overlapping areas. Because the images captured from the simulated environment are of a lower resolution than the originals, many of these logos are difficult to read in the simulated images; to aid with block discrimination, we assign a color to each block in addition to its logo. Table 6 reports the same statistics given in Bisk et al. [10] on the filtered subset of the data.

C.3 Dataset Ambiguity

A remaining limitation of the dataset is that some descriptions are ambiguous and not reliably actionable, since annotators did not attempt to execute the described actions. Some of the natural language descriptions are ambiguous, as depicted in Fig. 7.

C.4 Qualitative Error Analysis

Based on our results in Tables 4, we qualitatively examine some of the errors made by the Transformer-based model, where we observe several patterns. Looking at the validation examples with the highest IES, we observe that they often correspond to instances where the source block was mis-identified, leading the wrong block to be moved, and yielding a large IES. More interestingly, in the cases where the correct block was moved, the instruction often contains higher-order concepts as well as linguistic complexities such as ellipsis. For example, in Figure 6 there is a reference to a "disconnected square", which is a rare, higher-order geometric concept. In addition, the subsequent clause lacks an explicit reference back to the square; the annotator chose to leave this reference as implicit, given that the square is raised to a salient position by the previous clause. This type of noun-phrase ellipsis is common in natural language [50], and reflects the type of advanced pragmatic reasoning required to handle natural language.

Other examples of a higher-order concept instruction that the model performs poorly on are: "Take the Mercedes Benz block on the Burger King block without hiding the right top edge" (hiding), "targt [sic] goes 1/2 under Adidas with the right side hanging off" (hanging), and "take the Stella Artois block and place it on top of the Nvidia block, lined up perfectly" (perfectly). The model also struggles with long coreference chains, even when the anaphora are explicit, e.g. "Place the block that is to the right of the Starbucks block and make it the highest block on the board by placing it on the Mercedes block. It should be in line with the bottom block."

C.5 Full results for Experiment 2

Table 7: Full VGP results for the blocks dataset

Model	Embedding	Recon. Loss	Input	k	Block Acc.	IES
UNet	GloVe	Yes	State	1	55.3	3.3
UNet	GloVe	No	State	1	56.5	3.2
UNet	GloVe	Yes	Image	1	53.6	3.1
UNet	GloVe	No	Image	1	63.4	3.1
Transformer	GloVe	Yes	State	4	90.7	2.2
Transformer	GloVe	No	State	4	92.8	2.3
Transformer	GloVe	Yes	State	2	90.8	2.3
Transformer	GloVe	No	State	2	88.4	2.1
Transformer	BERT	Yes	State	4	88.9	2.6
Transformer	BERT	No	State	4	90.5	2.1
Transformer	BERT	Yes	State	2	89.0	2.4
Transformer	BERT	No	State	2	78.7	2.8
Transformer	GloVe	Yes	Image	4	88.9	3.4
Transformer	GloVe	No	Image	4	85.5	3.6
Transformer	GloVe	Yes	Image	2	79.3	3.2
Transformer	GloVe	No	Image	2	88.7	2.8
Transformer	BERT	Yes	Image	4	89.5	3.5
Transformer	BERT	No	Image	4	72.1	3.8
Transformer	BERT	Yes	Image	2	90.1	3.3
Transformer	BERT	No	Image	2	83.7	3.1

Table 8: Full VGP results for the blocks dataset on the physically feasible subset.

Model	Embedding	Recon. Loss	Input	k	Block Acc.	IES
UNet	GloVe	Yes	State	1	47.8	3.1
UNet	GloVe	No	State	1	49.1	3.1
UNet	GloVe	Yes	Image	1	45.5	3.0
UNet	GloVe	No	Image	1	56.4	3.0
Transformer	GloVe	Yes	State	4	88.2	2.4
Transformer	GloVe	No	State	4	90.3	2.4
Transformer	GloVe	Yes	State	2	89.1	2.6
Transformer	GloVe	No	State	2	85.1	2.4
Transformer	BERT	Yes	State	4	86.8	2.7
Transformer	BERT	No	State	4	88.3	2.2
Transformer	BERT	Yes	State	2	88.4	2.6
Transformer	BERT	No	State	2	72.0	3.0
Transformer	GloVe	Yes	Image	4	86.3	3.4
Transformer	GloVe	No	Image	4	83.0	3.7
Transformer	GloVe	Yes	Image	2	75.7	3.2
Transformer	GloVe	No	Image	2	84.4	3.0
Transformer	BERT	Yes	Image	4	88.3	3.7
Transformer	BERT	No	Image	4	66.2	3.7
Transformer	BERT	Yes	Image	2	88.5	3.4
Transformer	BERT	No	Image	2	86.3	3.5