Supplementary Material for Instruction-driven history-aware policies for robotic manipulations

Pierre-Louis Guhur[1], **Shizhe Chen**[1], **Ricardo Garcia**[1], **Makarand Tapaswi**[2], **Ivan Laptev**[1] **Cordelia Schmid**[1],
[1]Inria, École normale supérieure, CNRS, PSL Research University [2]IIIT Hyderabad
https://guhur.github.io/hiveformer/

In this supplementary material, we first present details about the model architecture in Section 1. Then we describe our categorization for RLBench tasks in Section 2 and experimental details in Section 3. We further provide additional ablations in Section 4. Finally, we conduct real robot experiments in Section 5 and present both quantitative and qualitative results on the real robot.

1 Model Architecture

UNet encoder for image encoding. The CNN in Eq (2) of the main paper is composed of 6 convolutional layers. The first two layers use kernels of size 3x3, strides of size 1 and output channels of sizes 8 and 16 respectively with LeakyReLU activation function. The remaining four layers use 3x3 kernels, strides of size 2 and output channels of size 16 followed by group normalization and LeakyReLU activation function. Therefore, an image of size $H \times W \times 3$ is encoded by a feature map of size $\frac{H}{16} \times \frac{W}{16} \times 16$.

UNet decoder for position prediction. The decoder uses a sequence of convolutional and upsampling layers to generate a heatmap on the point clouds. Specifically, the convolutional layer is fed with the output from the previous layer and the residual connection from corresponding layer in the UNet encoder. Its output channel size is 16, kernel size is 3 and stride size is 1. The upsampling layer uses scale factor of 2 and bilinear sampling. We stack 4 blocks of the layers to recover the original image size $H \times W$.

CNN for rotation and gripper. The CNN is composed of two convolutional layers, each with a 3x3 kernel, a stride of 2 and an output channel of size 64 followed by the group normalization and a LeakyReLU activation function. Then we apply average pooling and feed the flattened vector into a feedforward network to regress a 5-dimensional vector composed of 4-dimensional quaternion q_{t+1} and 1-dimensional gripper state c_{t+1}.

2 Categorization of RLBench Tasks

We evaluate on 74 available RLBench tasks. Although RLBench[1] currently contains 106 supported tasks, we had difficulties to produce demonstrations for 32 of them due to issues with the scripts and the motion planner. To analyze the performance of our model applied to different types of tasks, we manually group 74 tasks into 9 categories according to their key challenges. The 9 task groups are defined as follows:

- The **Planning** group contains tasks with multiple sub-goals (*e.g.* picking a basket ball and then throwing the ball). The included tasks are: basketball in hoop, put rubbish in bin, meat off grill, meat on grill, change channel, tv on, tower3, push buttons, stack wine.
- The **Tools** group is a special case of planning where a robot must grasp an object to interact with the target object. The included tasks are: slide block to target, reach and drag, take frame

[1]https://github.com/stepjam/RLBench/tree/master/rlbench/tasks

off hanger, water plants, hang frame on hanger, scoop with spatula, place hanger on rack, move hanger, sweep to dustpan, take plate off colored dish rack, screw nail.

- The **Long term** group requires more than 10 macro-steps to be completed. The included tasks are: wipe desk, stack blocks, take shoes out of box, slide cabinet open and place cups.
- The **Rotation-invariant** group can be solved without changes in the gripper rotation. The included tasks are: reach target, push button, lamp on, lamp off, push buttons, pick and lift, take lid off saucepan.
- The **Motion planner** group requires precise grasping. As observed in [1] such tasks often fail due to the motion planner. The included tasks are: toilet seat down, close laptop lid, open box, open drawer, close drawer, close box, phone on base, toilet seat up, put books on bookshelf.
- The **Multimodal** group can have multiple possible trajectories to solve a task due to a large affordance area of the target object (*e.g.* the edge of a cup). The included tasks are: pick up cup, turn tap, lift numbered block, beat the buzz, stack cups.
- The **Precision** group involves precise object manipulation. The included tasks are: take usb out of computer, play jenga, insert onto square peg, take umbrella out of umbrella stand, insert usb in computer, straighten rope, pick and lift small, put knife on chopping board, place shape in shape sorter, take toilet roll off stand, put umbrella in umbrella stand, setup checkers.
- The **Screw** group requires screwing an object. The included tasks are: turn oven on, change clock, open window, open wine bottle.
- The **Visual Occlusion** group involves tasks with large objects and thus there are occlusions from certain views. The included tasks are: close microwave, close fridge, close grill, open grill, unplug charger, press switch, take money out safe, open microwave, put money in safe, open door, close door, open fridge, open oven, plug charger in power supply.

3 Experimental Details

Motion Planner. We modified the default motion planner in RLBench, as it sometimes fails to reach a target pose even though there exist successful trajectories in the 3D space. To reduce the impact of the imperfect motion planner, we run the motion planner with different seeds up to 10 times until it finds a trajectory to the target.

Task Setup in Multi-variation Setting. For the multi-variation setting we choose tasks with as many variations as possible. We hence select the push buttons and tower tasks, for which we can easily construct new variations, as illustrated in Figure 1. For each of these tasks we use 100 variations for training and 100 different variations for testing.

- The **Push Buttons** task has three buttons with unique colors in the scene. The robot should press some or all of the buttons according to the order in an instruction. Variations of the task are defined by the different order and different colors of buttons. RLBench provides three sentence templates to generate synthetic instructions with changing button colors such as "push the red button, and then push the cyan one".
- The **Tower** task is inspired by the "stack block" task. The robot must stack some of the three colored cubes at a target location following the color order provided by the instruction. We generate synthetic instructions for each variation, such as "Stack the red, blue, green blocks", or "Stack the yellow block. Stack the purple block on top of it, then add the cyan cube".

Collection of human-written instructions. In addition to synthetic instructions used for training, we collect human-written natural language instructions for testing. We collect 162 human-written instructions and measure the success rate for each instruction on 10 episodes with random object locations. 8 native English speakers participated in the dataset collection, leading to 63 instructions of 51 testing variations for the push buttons task, and 99 instructions for 99 testing variations for the tower task. Human-written instructions are more varied than the synthetic ones. They contain unseen verbs (e.g. "Tap on the green button, then the grey button and end up pressing the pink button"), unseen formulations (e.g. "Press the green, cyan and pink buttons in that order"), longer sentences (e.g. "Press the white button and then you go to green button and press it and finally press the black button") or unseen color references (e.g. "Press the darker blue button, then the gray one and finally the lighter blue button.").

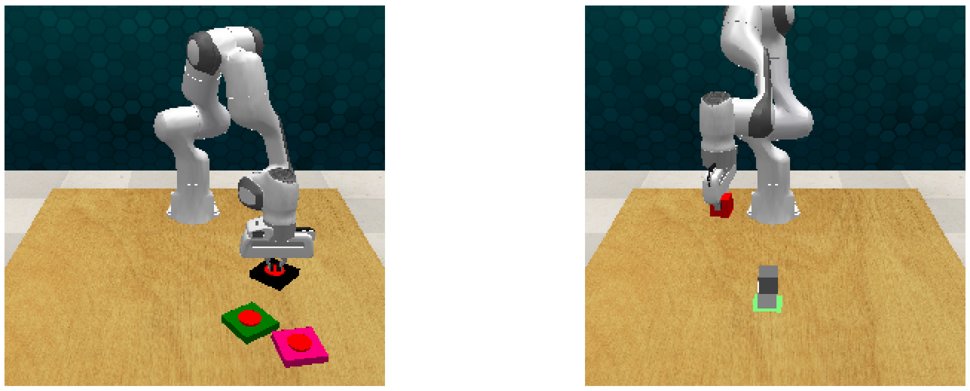


Figure 1: The tasks used in multi-variation setting. Left: push buttons task. Right: tower task.

Table 1: Comparison with LanCon-Learn [2] on 10 tasks.

	Hist.	Pick & Lift	Pick-Up Cup	Push Button	Put Knife	Put Money	Reach Target	Slide Block	Stack Wine	Take Money	Take Umbrella	Avg.
Single-task learning												
LanCon-Learn	-	20.2	25.2	96.2	57.8	91.4	99.6	60.2	57.0	58.4	73.0	63.9
LanCon-Learn	✓	64.8	56.8	96.4	59.4	90.6	98.7	63.4	56.6	67.8	74.8	72.9
Ours	✓	92.2	77.1	99.6	69.7	96.2	100.0	95.4	81.9	82.1	90.1	88.4
Multi-task learning												
LanCon-Learn	-	18.2	23.2	80.2	28.8	59.6	100.0	38.8	25.2	58.2	45.6	47.8
LanCon-Learn	✓	52.6	44.2	81.5	32.2	75.6	100.0	42.2	24.6	70.2	50.8	57.4
Ours	✓	88.9	92.9	100.0	75.3	58.2	100.0	78.7	71.2	79.1	89.2	83.3

4 Experiments on the simulator RLBench

We conducted further ablation studies to confirm our approach.

4.1 Comparison with Additional State-of-the-Art Approach

LanCon-Learn [2] is a recent instruction-conditioned multi-task approach. It takes as input the gripper state and the object state, namely the ground truth pose of each object in the scene, instead of raw visual observations as ours. It encodes instructions with GloVe embeddings and a bi-directional LSTM. It predicts the next pose of the gripper based on a modular architecture conditioned on encoded text features. We run experiments on RLBench with the code provided by the authors. Since some RLBench tasks require identifying the colors of an object, we modify their object state to include a RGB reference of each object. Moreover, we complete their method with a history mechanism, where the gripper state is concatenated with the gripper state from the previous step.

As described in Table 1, we obtained an average success rate of 63.9% with their original method (vs. 72.9% with history vs. 88.3% for our approach) in our single-task setting and 47.8% (vs. 57.4% with history vs. 83.8% for our approach) in our multi-task setting.

4.2 Additional Ablations on Multi-variation Setting

In the multi-variation setting in Table 2, the gap between LanCon-Learn and our approach is more significant than in Table 1, since GloVe embeddings differentiate poorly colors: for the "push buttons" task on unseen variations and synthetic instructions, the performance reaches only 1.7% (vs. 86.3% with our approach). This also happens when replacing CLIP embeddings with BERT in our model (40.2%).

The important role of history for long-term planning tasks such as "pushing buttons" is confirmed when comparing LanCon-Learn or our model with and without history in the Table 2. The model without history can only use its current observation to predict the next action. Therefore, it is hard to infer which buttons have been pressed and which button is the next target, leading to poor performance on the task.

Table 2: Comparison with LanCon-Learn [2] and ablation of the instruction encoding in the multi-variation setting for seen or unseen variations and synthetic, corrupted or real instructions.

Method	Hist.	Instructions Format	Instructions Encoder	Instructions Emb. E_T^x	Visual Emb. E_T^v	Push buttons Seen Synt.	Push buttons Unseen Synt.	Push buttons Unseen Corr.	Push buttons Unseen Real	Tower Seen Synt.	Tower Unseen Synt.	Tower Unseen Corr.	Tower Unseen Real
LanCon-Learn	No	Cat.	GloVe	-	-	25.6	12.1	0.3	0.1	21.3	9.1	0.1	0.0
LanCon-Learn	Yes	Cat.	GloVe	-	-	37.8	16.7	1.6	0.9	34.9	14.2	1.2	0.8
Ours	No	Seq.	CLIP	✓	✓	8.6	3.6	0.3	0.1	7.1	4.5	0.2	0.0
Ours	Yes	Avg.	CLIP	✓	✓	9.1	1.1	0.0	0.0	5.3	0.2	0.0	0.0
Ours	Yes	Seq.	CLIP	-	✓	100	83.2	81.1	71.3	77.1	53.2	51.3	21.3
Ours	Yes	Seq.	CLIP	-	-	86.2	65.2	56.4	49.8	54.9	34.8	29.8	24.7
Ours	Yes	Seq.	BERT	✓	✓	54.6	40.2	15.6	21.8	42.9	28.9	8.2	10.2
Ours	Yes	Seq.	OHE	✓	✓	100	3.1	0.1	0.0	96.8	3.8	0.4	0.0
Ours	Yes	Seq.	CLIP	✓	✓	100	86.3	85.6	74.2	77.4	56.2	53.6	24.1

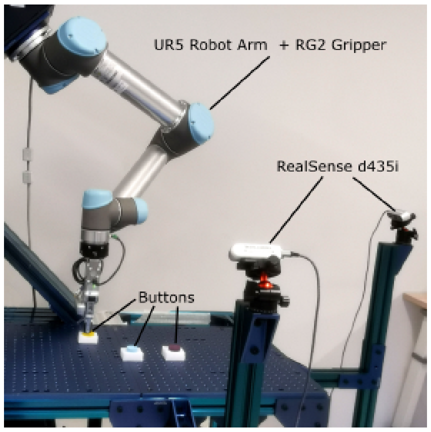


Figure 2: The robot scene with two RGB-D cameras and a UR5 robotics arm with an RG2 gripper.

We found that removing E_T^x decreases the performance only by 3.1%, but removing both E_T^x and E_T^v decreases the performance by 21.1%. Moreover, replacing the instructions with one-hot encoding of the variation index increases the performance for seen variations (by 19.4% on the tower task), but prevents the model from generalizing to unseen variations.

We performed ablations with corrupted instructions on unseen variations. Corrupted instructions were created from synthetic instructions by replacing color references with synonyms that have not been seen during training. For example, the color "azure" is replaced with "light blue", and the color "maroon" with "dark red". Baselines using CLIP as a language encoder have a much smaller drop of performance than any other encoder.

We also test our model with a global language embedding (average over word tokens) as in [3] and observe a significant drop in performance. The main reason is that the averaged embeddings do not represent well different action orders, *e.g.* we have obtained the average cosine similarity of 0.97 for instructions corresponding to same actions in different orders.

5 Experiments on Real-robot

Setup details. The cameras are Intel RealSense RGB-D cameras mounted on a fixed support as illustrated in Figure 2. We adapt our model to use $K = 2$ cameras. The resolution of the captured images is at a resolution of 1280×720, we apply center crop and downsampling to obtain images of size 128×128, which is the input to our model. We use nearest approximation to downsample depth images and bilinear approximation for RGB images. We use intrinsic parameters provided by Intel, and perform extrinsic calibration between the camera and the robot base-frame using an AprilTag marker [4]. We built 10 buttons using white cellulose foams: we manually cut them into 5×5 cm squares and attached to each square a painted rounded foam. The button bases and buttons have an average size of 4.95 ± 0.1 cm and 3.18 ± 0.22 cm respectively.

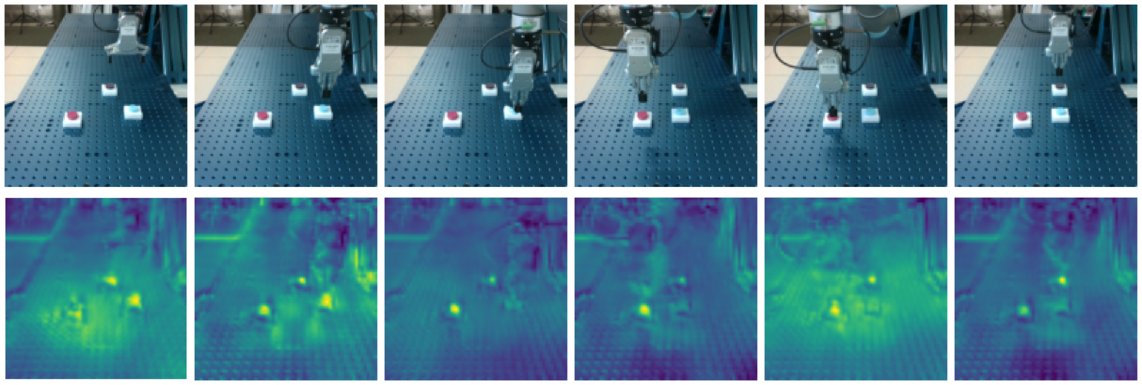


Figure 3: The instruction is "Press the cyan button, and then press the rose one, and then press the purple one". Top row: sequence of observations from one of the two side cameras in the robot scene. Bottom row: sequence of predicted attention maps by our model that indicate the gripper's position for the next step.

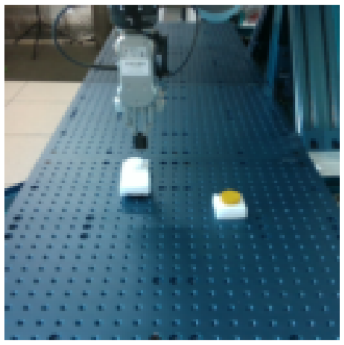
To collect demonstrations with the real robot, we design a script that automatically solves the task provided ground truth locations of buttons and the correct sequence of actions. In each demonstration objects are placed at random locations on the workspace and actions are executed at 10 Hz. We finetune the model for 8k iterations using the same training setup as that in the simulator.

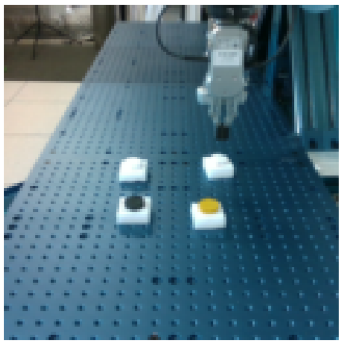
Qualitative Results. Figure 3 shows a successful example from our real robot experiments. The attention maps reveal that the robot correctly attends to the next buttons. Thanks to the history of previous observations and actions, the model is confident to not press a button that has already been pressed before (for example the cyan in the fourth column).

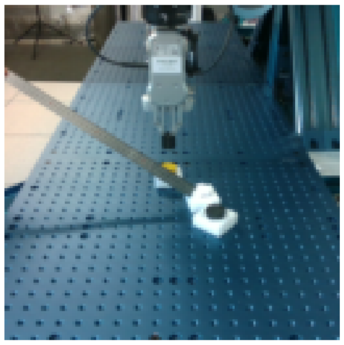
In Figure 4, we analyze the robustness of our model for to unseen variations in more challenging situations. The instruction of the variation is written by human: *"Press the yellow button and then press the black button and finish with the white button"*. Our model successfully pressed the buttons in the correct order for all situations in Figure 4.

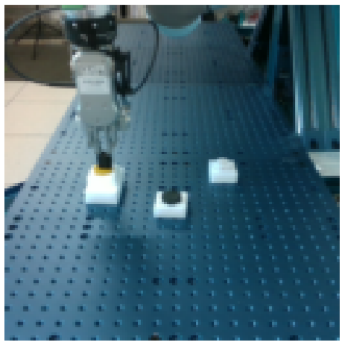
- Figure 4a: Since the foam buttons have low friction with the table, the gripper has accidentally flipped the black button, providing two buttons looking white. However, thanks to its history component, the robot is able to successfully press the right white button instead of the flipped button.
- Figure 4b: Two white buttons are present in the scene. This is a multi-modal example, in which the robot might predict a mean position between the two white buttons, whereas our robot can cope with this challenge.
- Figure 4c: We use a ruler to move the location the white button in the scene after the robot pushed the yellow button. Although such perturbations have never been used in training sequences, the robot remains robust to this dynamic environment.
- Figure 4d: We change the shape of the button by increasing the height of the yellow button.
- Figure 4e: We add occlusion to the gripper.
- Figure 4f: We change the appearance of the table. Our model is robust to the above visual modifications.

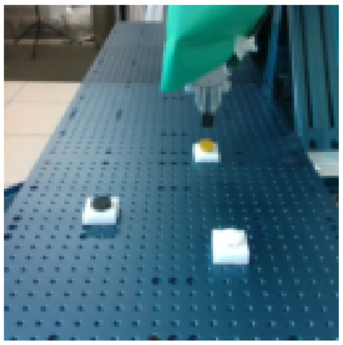
More details and video demonstrations of our real-robot experiments are available from the project webpage [5].


(a) Flipped button.


(b) Multimodal.


(c) Dynamic Environment.


(d) Unseen Button.


(e) Unseen Gripper.

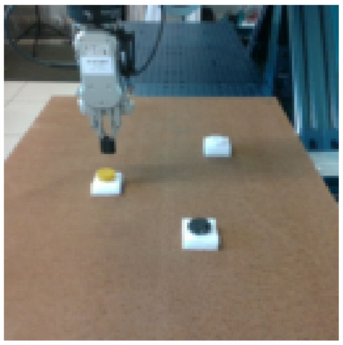

(f) Unseen Table.

Figure 4: Robustness of the learned policy on an unseen variation: *"Press the yellow button and then press the black button and finish with the white button"*.

References

[1] S. James and P. Abbeel. Coarse-to-fine Q-Attention with learned path ranking. *arXiv preprint arXiv:2204.01571*, 2022.

[2] A. Silva, N. Moorman, W. Silva, Z. Zaidi, N. Gopalan, and M. Gombolay. Lancon-learn: Learning with language to enable generalization in multi-task manipulation. *IEEE Robotics and Automation Letters*, 7(2):1635–1642, 2021.

[3] M. Shridhar, L. Manuelli, and D. Fox. CLIPort: What and where pathways for robotic manipulation. In *CoRL*, pages 894–906. PMLR, 2022.

[4] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *ICRA*, 2011.

[5] Project webpage. `https://guhur.github.io/hiveformer/`.