In this supplementary material, we provide further details, experiments, and descriptions of the attached media, to reinforce the results and conclusions from the main body of our paper. For a more fluid viewing experience please look through our project website, where videos (and corresponding descriptions) are side-by-side: https://sites.google.com/view/lila-corl21. The top-level page contains videos from our actual user study, showing the various models in practice, while the subpages contain additional experiments exploring the poor performance of the imitation learning baseline in our work, as well as justifications for the omission of the no-language latent actions baseline.

We additionally provide a full version of our code repository at the following url: https://github.com/siddk/lila, with a detailed README spanning the entire LILA pipeline from demonstration recording to training models, to deploying them on a robot.

A COVID-19 Impact Statement

Due to the ongoing COVID-19 Pandemic, the ability to run larger-scale user studies, and even have reliable lab access for preparing experiments, was limited. We are in a University setting, and as such, were susceptible to University restrictions.

All members of the authorship team had to go through a COVID-19 safety training, frequent testing, as well as an official approval process to be granted permission to work in the robot lab. For User Study participants, we were mostly limited to those with pre-existing access to the Engineering Building (spanning both robotics and non-robotics students), as well as a limited number of other University students granted approval to access other nearby buildings. This limited our ability to launch a larger scale user study, with more than 10 participants from a more diverse population.

That being said, we strongly believe that our existing participant statistics – 10 students (5 female/5 male, age range 23.2 ± 1.87 – reflect a broader user pool. Coupled with the statistical significance of the results we have already collected, we feel that the User Study results remain compelling, and our conclusions hold. That being said, we would like to run additional studies once COVID-19 restrictions relax in our area.

B Related Work Discussion

The main body of our paper contains a cursory discussion of different approaches for language-informed robotics, spanning a multitude of full-autonomy solutions. While this discussion helps provide contrast for our *shared-autonomy & language* based approach, LILA, it does not do the existing work in the field justice, nor does it discuss the data, implementation, and feature-engineering considerations that are coupled with these different approaches.

First, we discuss work that leverages structured logical forms as an intermediate representation for mapping language. The benefits of these forms are that they induce *logical forms* – functional, often programmatic – representations of meaning, that can then be executed on a robot, either by directly formulating a plan (requiring a model of the environment), or learning a lightweight policy from data that can fulfill these logical forms. A perceived benefit of these types of approaches is their sample efficiency – by leveraging a highly structured logical form and possibly hand-engineered features, one can learn the language to logical form mapping with 10s of examples. An example of this work is MacGlashan et al. [1] that learns reward functions given hand-engineered linguistic features; however, these reward functions are fed to a planner (requiring full knowledge of world dynamics – a huge assumption) to generate robot behavior. Follow-up work by Arumugam et al. [2] relax the hand-engineered language feature assumption by using more recent neural approaches, but still hinge on using planners. While these approaches are sample-efficient, many of the assumptions around planning and full dynamics are strong, and limit the potential of scaling this work (and for manipulation, are not necessarily straightforward!). Other work that relaxes the planning/dynamics assumption is Duvallet et al. [3]; this work uses hand-engineered features on top of a popular logical form – Spatial Description Clauses (SDCs) – to learn policies for robot navigation. While seemingly as sample efficient as the prior approaches without the downsides, there is still a high cost for

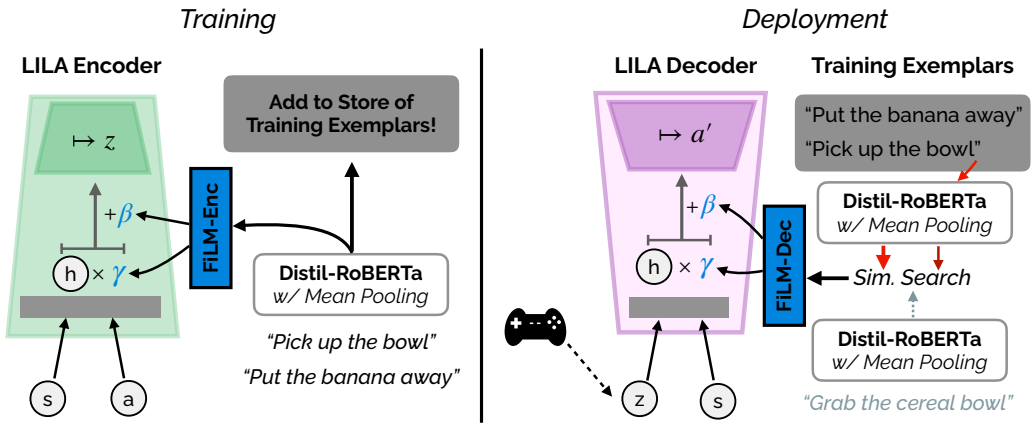

Figure 1: An overview of our language-informed latent actions (LILA) models with the FiLM modules highlighted. For additional clarity, we provide partial views of both the Encoder (during training) and Decoder (during deployment).

policy learning. Though the approach only required 10s of examples to learn to map language to the appropriate SDC, learning an effective policy (note this is just discrete node navigation – not continuous manipulation) required running DAgger [4] for 25 iterations on top of their existing data, collecting an order of magnitude more demonstration data (or demonstration edits/corrections as in DAgger) than originally given – 100s - 1000s of demonstrations.

On the other end of the spectrum are more recent approaches in end-to-end robot learning for more complicated tasks spanning navigation [5, 6] and manipulation [7]. This work is done in simulation, where one has the ability to do virtually infinite policy rollouts given a language instruction paired with a reward function, to learn robust policies via reinforcement learning. Other work in this paradigm that uses imitation learning, but with real-world deployments include works for quadcopter flight [8] and some limited manipulation [9]. These works still require 1000s of demonstrations, but use smart tricks for data augmentation and synthetic generation to learn robust policies.

LILA hopes to fill a void between these two classes of approaches; retaining the sample-efficiency of the earlier approaches, without the need for hand-engineered features, strong assumptions about known dynamics, or limited generalization potential. The experiments in the main body show that LILA is *extremely* sample efficient; however, the scope of this work is mostly using language as a means for *disambiguation*. It is our ardent hope that future work in language & latent actions (and shared autonomy more generally) turns to more dynamic settings, richer language, and hard forms of generalization; this work is just the first step.

Finally, we want to help fill out the story of language-informed robotics with work that does not necessarily involve *execution* (learning a policy or control space for robots to follow language instructions), but that can help language-based methods generalize better, from less data, or that leverage other modalities to help with specification. For example, Matuszek et al. [10] learn to map *unscripted* interactions consisting of language and gestures to object localizations from relatively few interactions; such methods are crucial for scaling up LILA to multiple objects, referring expressions, and compositional language instructions. Other work looks at other modalities like speech and gesture to learn logical forms that allow for efficient generalization [11]. Other work combines several modalities on top of language like gaze, gesture, and intonation to further help lift representations of human intent from language [12, 13]. All this work, though not directly related to LILA in that they do not help learn meaningful control spaces, do present possible avenues through which we might scale this approach to new contexts, language instructions, and behaviors.

C Model Architectures & Training

Following from the main body of the paper, we provide additional details on the model architectures, and additional data processing/augmentation we use in this work. We start with a more thorough description of the feature-wise linear modulation (FiLM [14]) mechanism we use to integrate language into the latent actions pipeline.

FiLM Architecture for LILA. Prior work in latent actions [15, 16] implement the latent action models as simple feed-forward multi-layer perceptrons (MLPs) with `tanh` activations as the non-linearity between layers. The Encoder and Decoder are symmetrical, with the Encoder encoding a combination of (s, a) pairs down to the latent space z (via an intermediate layer or two), and the decoder decoding from (s, z) back up to the 7-DoF action a. The intermediate layers in both the Encoder and Decoder have the same dimensionality of 30 – we refer the reader to consult the attached code for more detail.

With LILA, the two differences are that 1) we use the `GELU` activation instead of the `tanh` due to its better stability, and 2) we incorporate language into this existing pipeline. Recall that we use a version of the pretrained Distil-RoBERTA language model [17, 18] to generate embeddings of each user utterance. These embeddings have dimensionality of 768, which far exceeds the dimensionality of the 7-DoF (s, a) of the typical pipeline. Initial experiments attempting to naively concatenate the language embedding with the 7-DoF states and actions (or states and z values, in the case of the decoder) were not fruitful, as we were unable to learn (loss failed to decrease when training).

Instead, we turned to FiLM. The core principle with FiLM is that it's a fusion mechanism that does not increase the intrinsic parameter count of the core neural network (e.g., the original Latent Action MLP described above). FiLM has found great success in many multi-modal tasks for this reason, integrating with pre-existing pipelines in image classification to enable visual-question answering [14], as well as integrating into existing pipelines for instruction following in reinforcement learning [19, 20]. FiLM works as follows: given an intermediate representation from the Encoder or Decoder h with dimensionality d (say after the first layer of the corresponding MLPs), and a language embedding e, FiLM works in the following fashion:

$$\begin{aligned} \text{FiLM-Gen}_\theta(e) &= \gamma_e, \beta_e \\ h' &= \gamma_e \odot h + \beta_e \end{aligned}$$

where h' is the representation fed to later layers in the Encoder/Decoder MLP, and $\odot$ denotes the Hadamard product (component-wise multiplication). Simply put, Film-Gen$_\theta$ is a module that learns to *shape* the representations learned by the core Latent Actions MLP, injecting language information through this affine transformation defined by γ_e, β_e. We implement Film-Gen$_\theta$ as a separate two-layer MLP that also uses the `GELU` activation. Fig. 1 breaks this down visually, showing how the FiLM modules are added for both the Encoder and Decoder.

Imitation Learning Architecture. For Imitation Learning, we do not have this Encoder-Decoder structure, with two separate FiLM modules. Instead, Imitation Learning is implemented as a single MLP (of same parameter count/number of layers as the Encoder + Decoder in LILA) that conditions on the state s; we add a single FiLM module after the first-layer of this MLP.

Data Augmentation. Of secondary importance is in how we perform data augmentation for training. There are two key aspects to our data augmentation procedure: 1) enforcing latent action consistency, and 2) adding robustness to noise.

A key desire in latent action models is consistency in nearby states – executing the same latent action z in nearby states should be roughly similar. More formally, $d_T(\mathcal{T}(s_1, \phi(z, l, s_1)), d_T(\mathcal{T}(s_2, \phi(z, l, s_2)))) < \epsilon$ for $\|s_1 - s_2\| < \delta$, for some $\epsilon, \delta > 0$, where d_T is some distance metric (e.g., Euclidean distance between states). We enforce this with a sliding window approach; given a sequence of states within a fixed window size, we train the decoder to predict the same actions given the (z, l, s) triples for all states within the window.

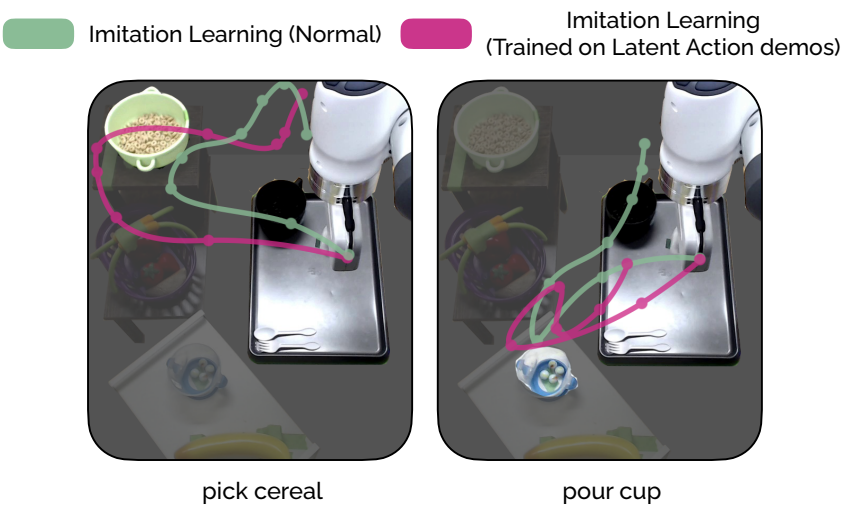

Figure 2: Visualized Trajectories for an Imitation Learning model trained on the "pure" data (that is reported in the paper), and an Imitation Learning model trained on the "LILA-style" demonstrations with the "sweeping" motions. On the left are trajectories for the instruction *"grab the cereal bowl"* and on the right, *"pour the blue cup into the black coffee mug"* – these two examples are from our training language set. We observe that Imitation Learning trained on the "LILA-style" demonstrations performs slightly worse by not following the complete ideal task trajectory, although neither approach is able to successfully complete the task.

The second, and most important augmentation we do is adding robustness to noise. Though we collect a dataset of (s, l, a) triples, we use a unique property of our control space to obtain better robustness: because our states are the actual joint states (let's call this q), and actions a are just joint velocities ($\dot{q}$), we can "re-compute" actions between two sequential states (s_1, s_2) by just taking the finite difference $a' = s_2 - s_1$. This allows us to do the following: add noise to each initial state s_i subject to $\epsilon = \mathcal{N}(0, \sigma)$ (we use $\sigma = 0.01$), then compute the "corresponding action" by taking $a' = s_{i+1} - (s_i + \epsilon)$. Adding noise in this way significantly helps make our models more robust to small variations in state space, and can be viewed as a simulated version of the DART paradigm [21] for noise-robust imitation learning.

We train LILA models with both the above augmentations. While the former augmentation mode is not directly applicable to Imitation Learning approaches, the second noise augmentation mode is – indeed, we find we have to triple the amount of such augmentations, in order to get even slightly meaningful behavior from Imitation Learning models.

D Demonstration Collection

Both Latent Action models and Imitation Learning models require a dataset of language paired with corresponding demonstrations to perform learning. As mentioned in the main body of the paper, we collect these demonstrations *kinesthetically*, manually moving the robot arm to complete specific tasks, recording the joint states and actions along the way. Critically, for imitation learning, these demonstrations consist solely of what we call the "forward", or "pure" demonstration of a task: starting at the home position, perform each of the individual subtasks smoothly and continuously, until the task has been satisfied. For a task like "put the banana in the fruit basket" this corresponds to 1) smoothly reaching for the banana and grasping it, 2) lifting the banana up and moving over to the basket, and 3) inserting the banana into the basket and releasing the gripper.

However, when collecting demonstrations for latent action models, we find that we can learn *better, more reliable* models by changing up the demonstration process a little, incorporating discrete segments of the demonstration where we *back off and then repeat* a motion. Concretely, for a task

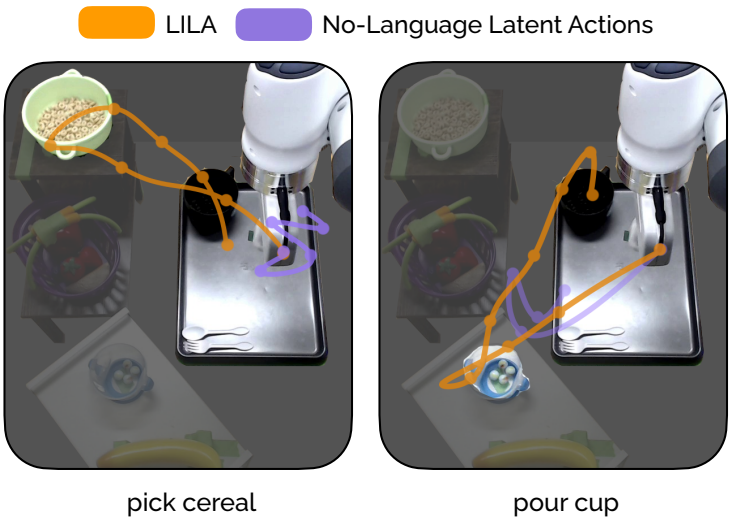

Figure 3: Visualized Trajectories for a No-Language Latent Actions model vs. LILA (with language) trained on the same set of demonstrations, operated by an expert user. With the No-Language Latent Actions model, the user tries their best to complete the task with the provided controls. On the left are trajectories for the instruction *"grab the cereal bowl"* and on the right, *"pour the blue cup into the black coffee mug"* – these two examples are from our training language set. We observe that the No-Language Latent Actions is unhelpful for completing the task, and unable to even completely reach the target object for either task, demonstrating the importance of incorporating language to help condition the learned latent actions.

like "put the banana in the fruit basket" this corresponds to 1) smoothly reaching for the banana, pulling back to the home position, *and then* reaching for the banana again and grasping it, 2) lifting the banana up, lowering it, then taking it to the basket, and 3) finally dropping the banana in the basket. These "sweeping" back-and-forth motions intuitively help the latent actions model induce control spaces that give users *reversible* control – the ability to move the robot back, rather than just press forward; this is critical to usability and recoverability.

On the Fairness of Comparing Imitation Learning and LILA. Because LILA and Imitation Learning are trained with different demonstrations, there is a plausible fear that our comparison between LILA and Imitation Learning is unfair – specifically, because the latent actions demonstration incorporates extra motion, LILA technically may be seeing more (s, a) pairs per demonstration compared to Imitation Learning, and can therefore learn better.

We mitigate this in two ways; first, as a side effect of doubling the number of collected demonstrations for Imitation Learning – recall that, in the main paper, to get IL to show semantically meaningful behavior, we needed to collect 30 demonstrations per task instead of the 15 per task LILA was given – we found that Imitation Learning sees 40% *more* (s, a) pairs than LILA (even more if you account for the fact that we tripled the data augmentation for Imitation Learning!). Second, Appendix E presents a more concrete experiment where we train the Imitation Learning model *on the LILA demonstrations*, and show that the resulting model performs worse than when trained on the standard IL dataset.

E Additional Experiments: Imitation Learning Baseline

A critical question from our user study has to do with the poor performance of the imitation learning baseline relative to both LILA and End-Effector control. This section explores additional ablation experiments as well as a technical argument for why imitation learning performs poorly – namely due to sample inefficiency, exacerbated by our real-robot setting.

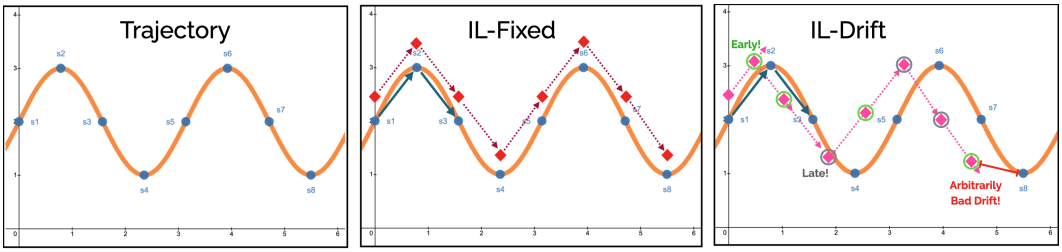

Figure 4: We consider a sinusoidal trajectory of a simplified end-effector through 2D space, where a robot's motion is continuous. If we can sample states when collecting this "demonstration" at a fixed interval, then the state-error of an imitation learning agent trained with behavior cloning grows minimally over time. However, the fixed interval assumption may not be true when collecting data on a real robot, where any noise can lead to compounding errors resulting in arbitrarily bad drift. This example supports our observation of poor IL performance on the 7-DoF Franka Emika Panda robot in our user study.

We also address the point raised in Appendix D – how Imitation Learning performs when trained with the "latent actions" style demonstrations ("sweeping" motions) rather than the "pure," straight-through demonstrations.

Ablation Experiments. The following URL contains several sets of ablation experiments, with corresponding text annotations: https://sites.google.com/view/lila-corl21/home/il-ablation. Many of these experiments show qualitative behavior, and are best viewed via the linked URL; however, we summarize the main findings here.

First, we put LILA and Imitation Learning on an equal footing, picking 3 of the 5 original tasks we used in the user study – namely, **`Pick Cereal`**, **`Pick Fruit Basket`**, **`Pour Cup`**. We collect 10 demonstrations for each task, and trained a base model with the noise-based data augmentation (add noise once for each state), for both LILA and Imitation Learning. We show that LILA is able to fully succeed at all three of these tasks with *only 10 demonstrations*, whereas Imitation Learning completely fails. However, we note that even at 10 demonstrations, Imitation Learning is starting to show semantically meaningful behavior – for the **`Pick Cereal`** the end-effector clearly moves toward the cereal bowl (though not close enough to grasp), and then down to the tray (though not close enough to execute a successful drop).

We then experiment with the impact of data augmentation, first looking at 3x the amount of augmentation (noising each state 3 times in the dataset following the procedure detailed above), and then 5x. We show that Imitation Learning performance is slightly better than the reference with 3x data augmentation, but that 5x doesn't additionally help.

We then vary the number of demonstrations from 10 to 20, then 30 demonstrations per task, with 3x data augmentation. Critically, we show that Imitation Learning *improves as we add more demonstrations, even though it still isn't able to solve the tasks.* As an interesting data point, at 20 demonstrations, the Imitation Learning agent is able to execute a successful grasp of the cup (see the video on the webpage), but cannot finish the task.

Unfortunately, after collecting 90 demonstrations (30 for each task), we felt we'd need close to 50-100 demonstrations per task to actually get imitation learning to solve these tasks – almost *two orders of magnitude* more data than LILA needed. This would have been prohibitive to collect so we stopped, and instead decided to analyze the possible cause of this extreme sample inefficiency. Again, videos and annotations depicting these ablations visually, in an easy-to-follow format can be found here: https://sites.google.com/view/lila-corl21/home/il-ablation.

Why is Imitation Learning Sample Inefficient? The above experiments point to an extreme sample inefficiency in Imitation Learning. Imitation Learning is generally subject to problems of cascading errors and the general noise problems of real robotics (imprecise resets, inherent noise in robot joints). However, we will now present an argument that illustrates why Imitation Learning

and LILA may be even worse than expected in our real-robot setting, due to specific implementation details in our publish/subscribe based methodology. Notably, there is imperfect communication between the top-level Python process (housing the learned models) and the low-level C++ robot controller that exacerbates the cascading errors imitation learning has to deal with. A graphical walkthrough of this argument can be found at the bottom of the webpage for this section here: https://sites.google.com/view/lila-corl21/home/il-ablation.

Consider the sinusoidal trajectory of a simplified end-effector through 2D space shown in Figure 4. The robot's motion is continuous, but we can sample states when collecting this "demonstration" at fixed intervals. Notably, this is an assumption present implicitly in most simulators (fixed frame rate, or fixed control iterations/sec) – this is also usually true when operating with discrete action spaces (move forward/right/left). However, this fixed interval assumption may not be true when collecting data on a real robot, depending on the implementation. More on this in a bit, but for now, let's assume our demonstration data consists of evenly spaced (state, action) pairs with this fixed interval between samples.

Consider training an imitation learning agent via behavioral cloning, where the policy is parameterized as a neural network. It's not clear what a NN will do given a state outside it's training distribution (could be arbitrarily bad) , but to simplify, *let's assume given a new state, this policy will predict an action based on retrieving the "nearest-neighbor" from it's training demo.* Assume there's some noise in the reset (this is representative – there's always *some* noise in the initial joint states - this is represented in simulators like Mujoco and PyBullet). If you roll out the imitation learning policy, you get behavior like that shown in the middle of Figure 4 – *critically, assuming the same constant sampling rate, the state-error grows minimally over time.* However, for continuous state and continuous action robotics grounded in a real-world robot, this assumption does not exist, which leads to the following point.

With our implementation on a 7-DoF Franka Emika Panda robot, we noticed that despite our best efforts, *we are not able to ensure states/actions are dispatched at a constant sampling rate*. The result (based on our straw man nearest-neighbors NN argument from above – though note the real world behavior, especially in higher dimensions will be much much worse!) is shown in Figure 4 on the right. With slippage in the read/publish times of states and actions, we can read states too early or too late, execute a "bad" action, and *cascade to arbitrarily bad final states over the course of execution* (even completely breaking in the middle of execution).

Imitation Learning with Latent Action Demonstrations. Finally, as raised in Appendix D, there is a question about the fairness of training on the "LILA-style" demonstrations with the "sweeping" motion vs. the more pure, forward-only imitation learning demonstrations. To address this, we train two Imitation Learning models (identical architecture, augmentation), with one model trained on the original "pure" data (the model from the main paper, used in the user studies) and a model trained on the "LILA-style" demonstrations with the sweeping behavior.

Fig. 2 shows visualizations of the trajectories for the models rolled out on two instructions from the training set. We see that the Imitation Learning model trained on the LILA demonstrations is worse than the "standard" Imitation Learning model. Whereas the pure model is able to at least closely reach the target objects, the other model attempts to reach for an object, but wastes time moving around the object rather than focusing on the task trajectory – intuitively this makes sense, as the "sweeping" motions present in the LILA data are confusing for imitation learning; given two opposing actions in the same state, what should it do?

These experiments, coupled with the experiments in the main paper provide ample evidence that the comparison between LILA and Imitation Learning is not only fair, but highlights the sample efficiency of LILA as well.

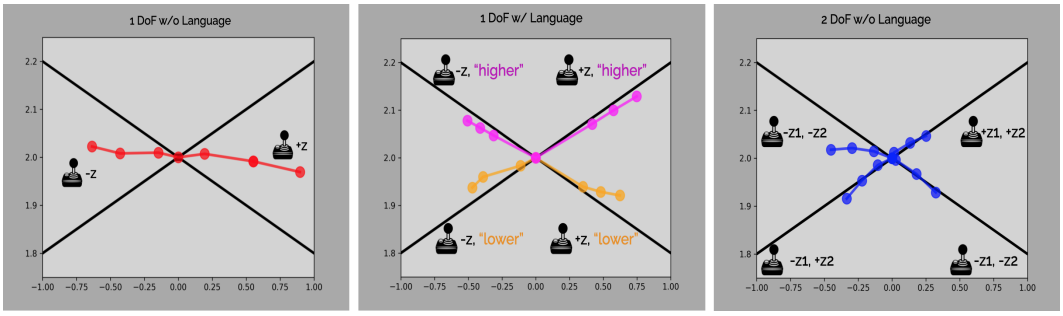

Figure 5: We train 3 different latent action models for the cross disambiguation task from Figure 2 of the main paper, which showed a simple "cross" example, where starting from the mid-point, the goal is to be able to navigate towards all 4 possible directions. As the standard LA framework only takes in the controller inputs, it is impossible for a user to succeed with only 1 degree-of-freedom (DoF). To solve this task, we need an additional axis to condition on (at least 2-DoF). Our results show that, as expected, without any additional input such as language, a 1-DoF controller (left) is incapable of task disambiguation, with a clear 0% task success rate for our simple cross setting.

F Additional Experiments: No-Language Ablation

Another possible question one might have is how latent actions performs on our tasks *without language* – in other words, is LILA necessary, or are prior latent actions models expressive enough to solve the tasks?

We answer this in two ways. First, we trained a latent actions model, completely ablating the language encoding pipeline (keeping architecture the same, but removing the FiLM components described in Appendix C). The corresponding latent action decoder only takes in the latent action z and state s as an input to predict high-DoF actions a. Fig. 3 shows visualizations of trajectories for this No-Language model as well as LILA operated by an expert user, trying to accomplish two specific tasks. While LILA performs as expected, the No-Language model is unable to make progress at all. As soon as control begins and the user moves the robot into a state close to any object, the control space loses meaning, and the user is unable to recover, instead generating random behavior. Again this makes sense; without language to condition on, the No-Language model has no idea what task to perform. It lacks the ability to *disambiguate* tasks, and as such, cannot make progress. We present additional videos and experiments to the above here: https://sites.google.com/view/lila-corl21/home/no-lang-baseline.

Second, we train 3 different latent action models for the cross disambiguation task seen in Figure 2 of the main paper: A 1 DoF without language baseline, LILA (our proposed approach) with 1 DoF, and a 2 DoF without language baseline. Each model is trained on a dataset of 100 demonstrations collected across the 4 tasks. We then visualize movement trajectories (see Fig. 5) by controlling the latent action (z for 1 DoF, z1 and z2 for 2 DoF). As expected, without any additional input such as language, a 1-DoF controller is incapable of task disambiguation, with a clear 0% task success rate for our simple cross setting. With both our 1-DoF controller w/ language input and a 2 DoF controller, task disambiguation is possible – highlighting the necessity of additional information of any modality. However, note that in large multi-task environments with dozens of tasks, re-designing a controller can be difficult – language is a much more flexible and natural way to add this information.

Together, these results motivate why no-language latent action baselines are incapable of being useful for multi-task environments, as they are limited by total degrees of freedom of the controller. Because the goal of our user study is to compare methods that could be useful for successful task completion, If included, such a baseline would be uninformative as it would be impossible for any user to achieve above a 0% task completion rate.

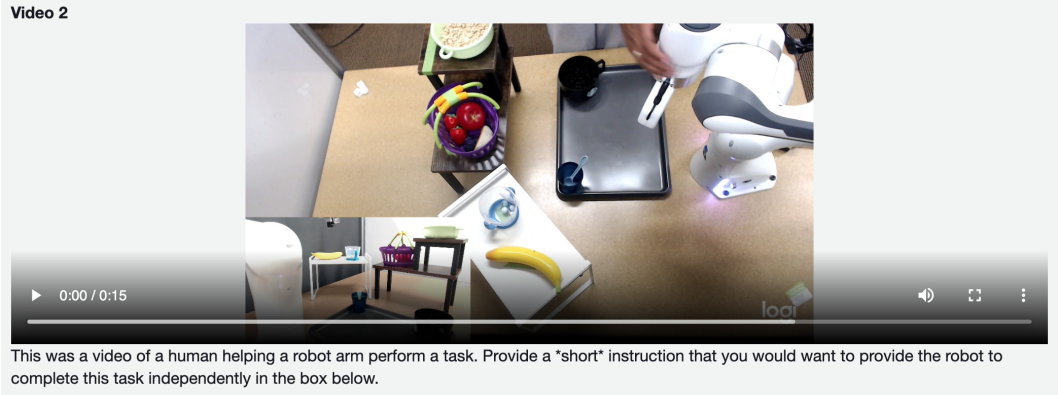

Figure 6: Interface shown to crowdworkers for collecting language utterances.

G Crowdsourcing & User Study

As described in the paper, to train LILA with language utterances, we hire crowdworkers on Amazon Mechanical Turk to provide language utterances give video demonstrations of a human assisting the robot arms. We paid crowdworkers 1.20 dollars to provide short utterances for $seven$ videos. Notably, crowdworkers were not given any information about the possible tasks, or names of the objects in the scene. An image of the interface provided to crowdworkers is included in Fig. 6.

Algorithm 1 Filtering Language Utterances

for $\texttt{task} = 1, 2, \ldots, T$ **do**
  Initialize list E_{task}.
  for $\texttt{user} = 1, 2, \ldots, N$ **do**
    Append embedding of utterance `embed(user, task)` to E_{task}
  end for
end for
for $\texttt{user} = 1, 2, \ldots, N$ **do**
  Initialize list D_{user}
  for $\texttt{task} = 1, 2, \ldots, T$ **do**
    Append `cosine distance` d between `embed(user, task)` and $\texttt{avg}(E_{\text{task}})$ to D_{user}
  end for
end for
Filter out all utterances from users with K highest $\texttt{avg}(D_{\text{user}})$

Filtering Crowdsourced Language Annotations. As described briefly in the main paper, one issue with crowdsourcing language utterances from Amazon Mechanical Turk is the presence of "spam", noise, or extremely out-of-domain annotations. We accept and pay all crowdworkers for their responses, but adopt the following filtering algorithm before training on the collected utterances:

We set the threshold K to use utterances from 15 different crowdworkers for each of the 5 tasks. Example utterances from crowdworkers who we filtered out include *"move your arm outwards towards the left about 10 inches, lower your arm until you can't anymore then close your claw."* and *"the human the robot to take bowl"*, highlighting the challenges of eliciting high quality language utterances from demonstrations, and the necessity of such filtering methods.

All 15 utterances for each of the 5 tasks are listed in the attached file `'full-annotations.txt'`, constituting the entirety of our training data. Furthermore, the file `'filtered.txt'` contains the utterances belonging to workers that were filtered out by our procedure.

User Study Instructions. As described in the main paper, we conducted a user study with 10 participants. The attached PDF file `'user-study.pdf'` shows the example instructions provided to each participant in our study.

References

[1] J. MacGlashan, M. Babes-Vroman, M. desJardins, M. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang. Grounding English commands to reward functions. In *Robotics: Science and Systems (RSS)*, 2015.

[2] D. Arumugam, S. Karamcheti, N. Gopalan, L. L. S. Wong, and S. Tellex. Accurately and efficiently interpreting human-robot instructions of varying granularities. In *Robotics: Science and Systems (RSS)*, 2017.

[3] F. Duvallet, T. Kollar, and A. Stentz. Imitation learning for natural language direction following through unknown environments. In *International Conference on Robotics and Automation (ICRA)*, pages 1047–1053, 2013.

[4] S. Ross, G. Gordon, and A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics (AISTATS)*, 2011.

[5] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[7] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] V. Blukis, N. Brukhim, A. Bennett, R. A. Knepper, and Y. Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *Robotics: Science and Systems (RSS)*, 2018.

[9] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor. Language-conditioned imitation learning for robot manipulation tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[10] C. Matuszek, L. Bo, L. Zettlemoyer, and D. Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2014.

[11] T. Kollar, J. Krishnamurthy, and G. P. Strimel. Toward interactive grounded language acqusition. In *Robotics: Science and Systems (RSS)*, 2013.

[12] C. Kennington, S. Kousidis, and D. Schlangen. Interpreting situated dialogue utterances: an update model that uses speech, gaze, and gesture information. In *SIGDIAL Conference*, 2013.

[13] D. Whitney, M. Eldon, J. G. Oberlin, and S. Tellex. Interpreting multimodal referring expressions in real time. In *International Conference on Robotics and Automation (ICRA)*, pages 3331–3338, 2016.

[14] E. Perez, F. Strub, H. D. Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[15] D. P. Losey, K. Srinivasan, A. Mandlekar, A. Garg, and D. Sadigh. Controlling assistive robots with learned latent actions. In *International Conference on Robotics and Automation (ICRA)*, pages 378–384, 2020.

[16] S. Karamcheti, A. Zhai, D. P. Losey, and D. Sadigh. Learning visually guided latent actions for assistive teleoperation. In *Learning for Dynamics & Control Conference (L4DC)*, 2021.

[17] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[18] N. Reimers and I. Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[19] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[20] D. Bahdanau, F. Hill, J. Leike, E. Hughes, S. A. Hosseini, P. Kohli, and E. Grefenstette. Learning to understand goal specifications by modelling reward. In *International Conference on Learning Representations (ICLR)*, 2019.

[21] M. Laskey, J. N. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning (CORL)*, 2017.