Joint Embedding Spaces for Language and Behavior

YuXuan Liu
Department of Electrical Engineering and Computer Science
University of California, Berkeley
`yuxuanliu@berkeley.edu`

Abstract

Designing well-shaped reward for reinforcement learning agents can be challenging, often requiring careful hand crafting and experimentation. We propose a method that interleaves reinforcement learning with grounded natural language in a way that is consistent with human feedback. We learn a joint embedding space between trajectories and language feedback that enables the agent to quickly adapt to human feedback by directly moving in the embedding space. Moreover, this embedding space can be used to explain differences in behavior using language.

1 Introduction

Specifying tasks for autonomous agents is often a challenging problem. In reinforcement learning, tasks are learned and defined in terms of a reward function. However, in many cases, specifying a reward function that is in line with desired behavior can be difficult.

Consider the task of autonomous driving, in which an autonomous car must from navigate from location A to destination B. What should be the reward for this task? Suppose we defined reward as a binary indicator for when the car reaches the destination. However, the agent may never experience any reward since it's extremely unlikely the agent will reach the destination by chance. If we include a shaping function that incentivizes the agent to be closer in Euclidean distance to the destination, the agent may learn to plow through pedestrians and drive off roads in the direction of the destination. Suppose we penalize the agent for injuring pedestrians and disobeying traffic laws, how should the agent learn to trade off injuring passengers in the car vs injuring pedestrians?

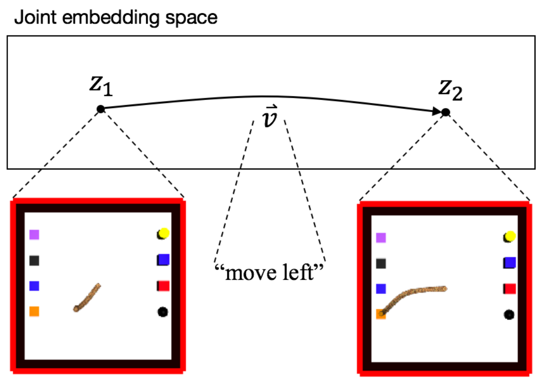

Figure 1: Joint embedding space for language and behavior: we construct a deep embedding space in which agent behavior, such a navigating to different locations, encode as points in the space z_1 and z_2. The language feedback describing the change in behavior is encoded as the vector between behavior points.

Even if we could enumerate all possible events and associate numerical weights to them, identifying when these events occur is an even harder problem. Avoiding pedestrians requires recognizing pedestrians in all types of traffic conditions, and following traffic rules requires encoding every rule into a format the agent understands. Each of these problems is a research problem in of itself.

Imitation learning is one approach to the reward design problem that learns from expert demonstrations. In traditional forms of imitation learning, the agent either learns a direct mapping of states to expert actions or infers a reward function in which the expert is optimal. However imitation is not the only way humans learn expert behavior. Rather, we can interpret natural language instructions such as in the form of a cooking recipe or a teacher's directions. These language instructions can describe dense task-specific rewards as well as provide online feedback for the learner. Suppose, in the navigation task, the autonomous car is given the task ”drive to the grocery store,” however it learns to make dangerous turns. The teacher could then provide feedback in the form of language such as ”slow down before turning” to guide the agent towards better behavior.

2 Related Work

Much of the related work in interleaving natural language with reinforcement learning has consisted of instruction following, in which a reinforcement learning agent is given a natural language instruction and the agent must take actions that realize the task specified in the instruction. Branavan et al. (2009) and Misra et al. (2017) learns policies conditioned on states and commands which are trained using reinforcement learning. Artzi and Zettlemoyer (2013) and She and Chai (2017) learn parsers over a grounded grammar to parse commands to actions. Mei et al. (2016) and Arumugam et al. (2017) learn recurrent neural network models that combine natural language instructions with agent observations to produce actions realizing those instructions. However, teaching an agent to follow language commands may not generalize well across new tasks, new environments, or new agents. Suppose the robot was taught to push orange and brown blocks, how should the robot generalize to pushing blue block when it has no grounding of what the blue block is?

Another direction of work maps learns natural language to functions in ways that are generalizable across environments or tasks. MacGlashan et al. (2015) and MacGlashan et al. (2014) takes a reward learning approach in which learned reward functions are generalized across to new environments and robots. Suddrey et al. (2017) and Branavan et al. (2012) learn high level planners and parses language instructions into low-level reusable modules. Andreas et al. (2017) breaks down larger tasks into reusable language-guided sub-tasks that can generalize to solving new tasks. Jänner et al. (2017) learns a universal value function conditioned on language instructions that generalizes across new instructions. Wang et al. (2016) learns mappings from language to sub-task commands in an interactive online game.

In terms of human feedback, many prior works learn policies from preferences. Christiano et al. (2017) learns a reward function that aligns with human feedback in an online reinforcement learning task. (Akrour et al., 2011, 2012, 2014) learn robust utility functions over user preferences for inferring rewards. Daniel et al. (2015) and MacGlashan et al. (2017) combine online reinforcement learning with preference ratings to simultaneously learn rewards while learning policies.

Learning deep embedding spaces have also shown remarkable ability in discovering structure in unsupervised datasets. Mikolov et al. (2013) learns distributed representations for words that capture contextual relationships among words. Kiros et al. (2015) extend distributed representations to the sentence level by considering sentence contexts. Bowman et al. (2015) learns a generative model over sentences using a variational lower bound. In reinforcement learning, Wang et al. (2017) learn a embedding space over trajectories and policies for robust imitation.

3 Learning Joint Embedding Spaces

In this work, we propose a method that interleaves language feedback with online reinforcement learning to learn policies that align with human preferences. We augment human preferences with language feedback annotations to learn a grounding from language feedback to the agent's behavior. We train the agent using human feedback in an online reinforcement learning system. In learning new tasks, the human can provide feedback directly in terms of the grounded language they used during annotation.

Our hypothesis is that while high-level language descriptions may not generalize well to new complex tasks, language feedback will be valid across tasks. Moreover, high-level instruction-following does not take into account incorrect interpretations of the instruction text while in our method, language feedback is conditioned on the agent's current behavior.

For instance consider the block pushing task, if the robot has no understanding of what a ”blue block” is, there is no means for the human to provide a high-level instruction that will complete the task. However, in our feedback method, if the robot was initially reaching too far below the blue block, we could provide language feedback such as ”reach higher” without without requiring

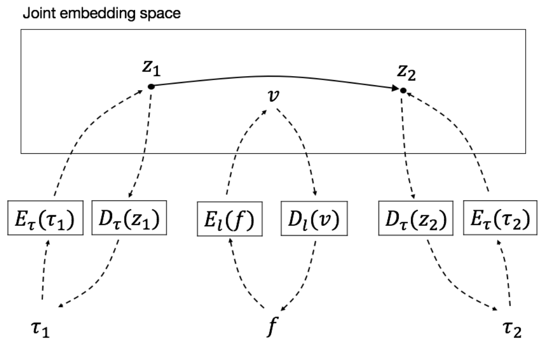

Figure 2: Joint embedding space model: behaviors τ_1 and τ_2 are encoded as points z_1 and z_2 in the space and language feedback f as the vector v between z_1 and z_2. Encoders $E_\mathcal{T}(\tau)$ and $E_l(f)$ map language and behavior as points and vectors in the space, and decoders $D_\mathcal{T}(z)$ and $D_l(v)$ map points and vectors from the space back into behavior and language.

the robot to generalize its understand of the "blue block."

To account for language feedback, we learn an embedding space in which language and behavior can be encoded together. Since, language feedback describes the difference between two behaviors, we can think of feedback as the language that takes you from behavior τ_1 to behavior τ_2. In the embedding space, we choose to map behaviors as points and language feedback as a vector between two behavior points. This joint representation of language and behavior enable us to build agents that can understand language feedback and describe changes in behavior using language.

To construct our deep embedding space, we learn a behavior encoder $E_\mathcal{T}(\tau) = z$ that encodes a behavior τ as a point z in the embedding space and language encoder $E_l(f)$ that encodes language feedback l as a vector v in the embedding space. To recover behaviors and language, we also learn a behavior decoder $D_\mathcal{T}(z) = \tau$ that can recover a behavior τ given a point z in the embedding space and a language decoder $D_l(v) = f$ that can recover language feedback f given a vector v in the space. Together, these encoders and decoders enable us to learn a well-structured space in which language feedback f describing two behaviors τ_1, τ_2 represents the vector between two behavior points in the embedding space:

$$E_\mathcal{T}(\tau_2) - E_\mathcal{T}(\tau_1) = E_l(f)$$

3.1 Model

While our model is agnostic to choices of representations for encoder, decoder, language and behavior, we will discuss possible choices for such representations and concrete implementation details.

First we consider behavior in the context of a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0)$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $\mathcal{T}(s_{t+1}|s_t, a_t)$ the transition distribution, and ρ_0 the initial state distribution. We consider finite T horizon trajectories with $s_0 \sim \rho_0$, $a_t \sim \pi_\theta(s_t|a_t)$, $s_t \sim \mathcal{T}(s_{t+1}|s_t, a_t)$ where $\pi_\theta(s|a)$ is a parameterized policy. A trajectory $\tau := \{s_0, s_1, ..., s_T\}$ is defined as a collection of states sampled according to the trajectory distribution induced by π_θ.

Since a policy π_θ induces a trajectory distribution $p_\theta(\tau)$, we can represent behavior in terms of the policy π_θ or samples from the trajectory distribution $\tau \sim p_\theta(\tau)$. For the purpose of simplicity, we chose to represent behavior in terms of trajectories. A model that predicts trajectories will not immediately provide a policy that can realize such trajectories; however, using imitation learning techniques we can subsequently learn policies to imitate predicted trajectories.

To encode trajectories, we parameterize $E_\mathcal{T}$ as a neural network. For long trajectories of arbitrary length, a bidirectional LSTM may serve as a good encoder for such state sequences. However, for the purpose of simplicity, we chose a fully connected neural network that encodes fixed-length trajectories. The input to the network is a flattened trajectory followed by two hidden layers with numbers of hidden units equal to half and a quarter of the input dimension respectively. We use Leaky-ReLU nonlinearities with an embedding space dimension of 32. The decoder $D_\mathcal{T}$ maps 32-dimensional behavior points into flattened trajectories with two hidden layers with numbers of hidden units equal to a quarter and half of the flattened trajectory dimension respectively.

Language feedback can be represented as a sequence of word tokens. One encoder and decoder representation for such a representation could be a seq-to-seq model where the encoders and decoders are recurrent neural networks. To avoid the complexity of such language models, we choose to represent feedback as a bag-of-words.

We parameterize the encoder E_l as a neural network with two hidden layers with numbers of hidden units equal to half the input dimension. The decoder D_l is a neural network with a single 32-unit hidden layer. The output activation is a softmax distribution over mutually exclusive groups of words.

3.2 Model Training

To train our model, we use several losses to ensure the embedding space is well structured. Moreover, we conduct an ablation study in Section 5, and show that all losses of the model are important in regularizing the model for generalization.

To ensure that language vectors in the embedding space correspond to differences in behavior points, we use an alignment loss:

$$\mathcal{L}_{\text{align}} = ||E_l(f) - (E_\tau(\tau_2) - E_\tau(\tau_1))||_2^2$$

This loss ensures that the encoded language vector $E_l(f)$ is close to the difference in behavior points $E_\tau(\tau_2) - E_\tau(\tau_1)$ in squared Euclidean distance.

To ensure that predicted trajectories are consistent with the language feedback, we use a prediction loss:

$$\mathcal{L}_{\text{pred}} = ||D_\tau(E_\tau(\tau_1) + E_l(f)) - \tau_2||_2^2$$

Here, the predicted trajectory of τ_1 after taking into account the feedback f, should be close to the ground truth trajectory τ_2.

For the language decoder to produce reasonable descriptions, we use a description loss:

$$\mathcal{L}_{\text{desc}} = CE(f, D_l(E_\tau(\tau_2) - E_\tau(\tau_1)))$$

Where CE is the cross entropy. This ensures that vectors between two behavior points decode to the correct ground truth language feedback.

We also regularize the trajectory encoder and decoder with the autoencoding loss:

$$\mathcal{L}_\tau = ||D_\tau(E_\tau(\tau)) - \tau||_2^2$$

This ensures that the encoder and decoder are producing consistent predictions.

For training, we optimize the total loss:

$$\mathcal{L} = \sum_{\tau_1, \tau_2, f} (\mathcal{L}_{\text{align}} + \lambda_1(\mathcal{L}_{\text{pred}} + \mathcal{L}_\tau) + \lambda_2 \mathcal{L}_{\text{desc}})$$

Where λ_1 and λ_2 are hyperparameters. We train the model end-to-end using the ADAM optimizer with learning rate 10^{-4}.

3.3 Learning with Language Feedback

Now we will consider three applications of our model: controlling the behavior of an agent using language feedback, extrapolation in the embedding space, and language descriptions of changes in behavior.

With a joint embedding space model, it is straightforward to adjust agent behavior using language. Suppose the agent currently exhibits behavior τ_0, and a human provides language feedback f to the agent. The model can encode the agent's behavior as a point $z_0 = E_\tau(\tau_0)$, encode the language feedback as a vector $v = E_l(f)$, and add z_0 and v to recover a new behavior point $z_1 = z_0 + v$. In the embedding space, z_1 represents the behavior τ_0 after accounting for language feedback. To recover the adjusted behavior, we can decode z_1 to behavior $\tau_1 = D_\tau(z_1)$ that the agent can follow.

Certain language instructions can also be applied repeatedly. Consider, for instance, the feedback instruction, "move right." If we want to tell the agent to move very far to the right, this amounts to telling the agent to "move right" several times. We can exploit the consistency in the embedding to achieve this. First, encode the agent's behavior as a point $z_0 = E_\tau(\tau_0)$, encode the language feedback as a vector $v = E_l(f)$. Since the feedback vector is constant for a given language instruction, we can apply it several times until the desired behavior is achieved. Namely, we can consider $z_1 = z_0 + \lambda v$ for $\lambda \in \mathbb{N}$. For well structured spaces, we can even consider $\lambda \in \mathbb{R}$ where $\lambda < 0$ could produce behavior opposite of the language feedback.

Explaining changes in agent behavior is important in building models and optimization methods that are interpretable. Using a joint embedding space model, we can use the language decoder D_l to recover descriptions of changes in behavior. Suppose we have to behaviors τ_1 and τ_2 with encodings $z_1 = E_\tau(\tau_1)$ and $z_2 = E_\tau(\tau_2)$. Since vectors in the embedding space represent language feedback, we can decode the vector between z_1 and z_2 to recover a description of the changes in behavior $D_l(z_2 - z_1)$.

4 Experiments

To evaluate the effectiveness of our method, we set up two experiments in simulation: 2D navigation and 2-DoF arm manipulation. In designing

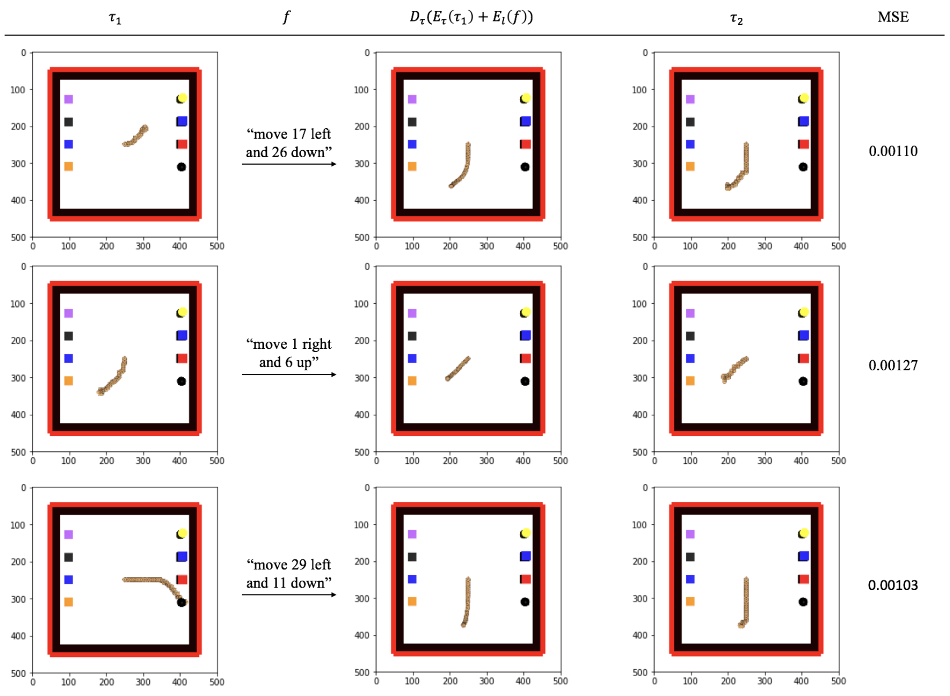

Figure 3: Qualitative and quantitative evaluation of behavior prediction on the 2D navigation task. For random behavior pairs τ_1 and τ_2 with language feedback f in the validation set, we show the predicted $D_\tau(E_\tau(\tau_1) + E_l(f))$ trajectory under our model as well as the MSE ($\mathcal{L}_{\text{pred}}$).

our experiments, we sought to address the following questions:

1. Can we adjust behaviors to language feedback using the learned embedding space?
2. Does extrapolation in the embedding space produce meaningful results?
3. Can the model offer reasonable explanations for differences in behaviors?

In the 2D navigation environment, the agent can move in the 4 cardinal directions and the state space is the position of the agent. We recorded trajectories of the agent moving to random target locations and randomly sampled two trajectories to describe the feedback that would take one trajectory to the other. While we had set up a manual annotation framework, it was more efficient to experiment with automatically annotated data for the purpose of our experiments. The feedback was labelled according differences in the final agent position, discretized to 60 bins along the x and y directions.

In the 2-DoF reaching environment, the robot is torque-controlled and trained to reach different locations on the plane. We recorded trajectories of the robot reaching to random target locations and collected pairs of trajectories that differed by a fixed amount along 8 directions spread uniform on the plane. The feedback was recorded as difference in the final end-effector position of the robot discretized to 3 bins along the x and y directions.

We collected 1000 pairs of trajectories and labels, and trained our model on 750 pairs with 250 pairs in the validation set.

4.1 Learning to Adjust Behavior

To evaluate our model's effectiveness in adjusting behavior according to language feedback, we evaluated our model's predictions on the validation set and generalization on examples outside the training set.

On the validation set, our model predicted translated trajectories with a mean squared error of 0.00376. Figure 3 compares this quantitative evaluation with a qualitative evaluation. Moreover, the model was able to generalize and predict behavior that wasn't seen in the training set as shown in Figure 5.

4.2 Extrapolation in the Space

For the 2-DoF reaching task, we found that our model could also adjust behavior in accordance to

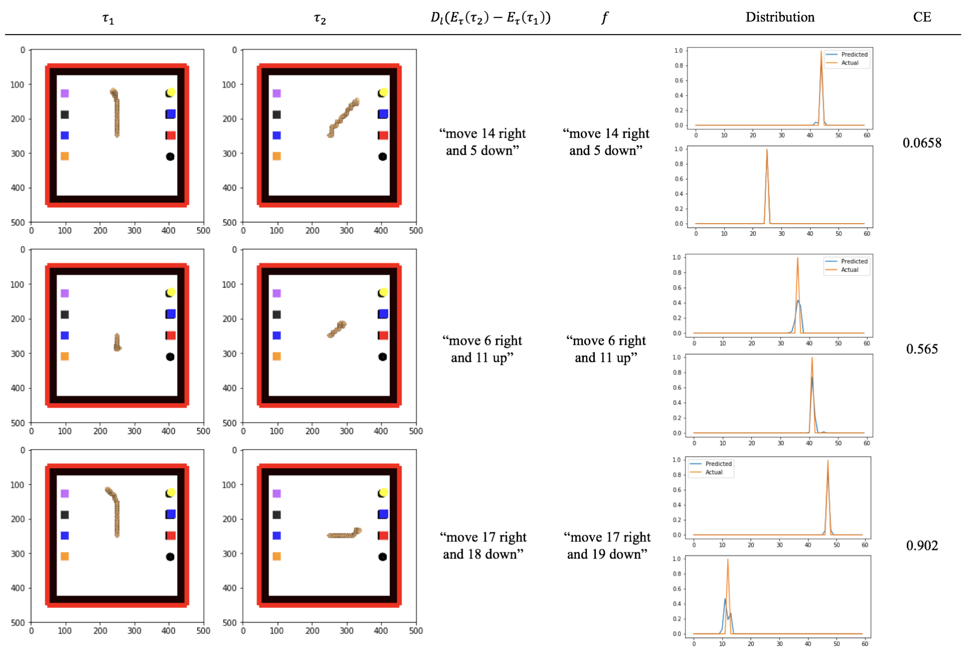

Figure 4: Qualitative and quantitative evaluation of behavior description on the 2D navigation task. For random behavior pairs τ_1 and τ_2 with language feedback f in the validation set, we show the predicted language description $D_l(E_\tau(\tau_2) - E_\tau(\tau_1))$ under our model, plot of the true one-hot word distribution vs the predicted distribution, and the average cross entropy between the true and predicted distributions ($\mathcal{L}_{\text{desc}}$).

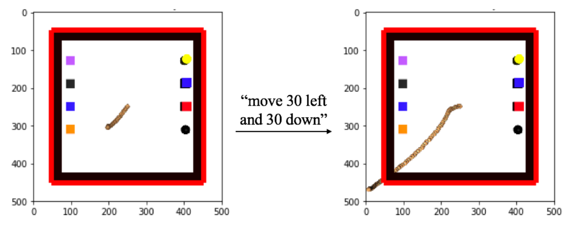

Figure 5: Generalization of our model on 2D navigation task. The model can incorporate language feedback that generalizes to states unseen in the training data.

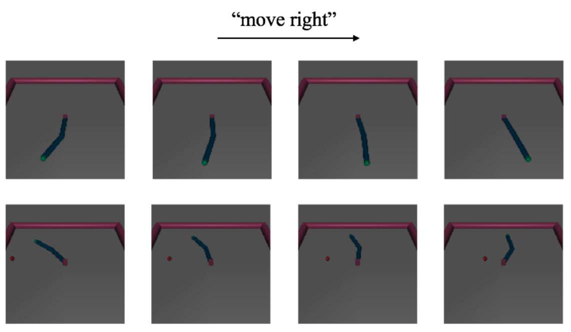

Figure 6: Extrapolation of behavior on the 2-DoF reaching task. We plot the final reaching location of the arm. We start with a reaching behavior that reaches towards the left-half of the plane and as we add multiples of the encoded "move right" feedback, the arm begins to reach towards the right-half of the plane.

language feedback. Since an encoding of a feedback text $E_l(f)$ is valid for any region of the embedding space, we can extrapolate in the direction of the feedback to get more extreme behavior.

We found that we can start with a behavior that reached to the left half of the plane and encode this behavior to a point z_1. Then, by adding multiples of the language feedback "move right," $z_1 = z_0 + \lambda E_l$("Move Right"), the reacher will slowly reach towards the right with increasing λ (Figure 6).

4.3 Describing changes in behavior

To evaluate our model's effectiveness in predicting language descriptions, we evaluate the language decoder on the 2D navigation task. For pairs of behaviors (τ_1, τ_2) in the validation set, we encode those behaviors into points z_1 and z_2 respectively. Then we decode the difference of those points us-

Ablation	Best MSE	CE	MSE	Best CE
Full model	**0.00376**	**1.15**	**0.00391**	**1.15**
Without $\mathcal{L}_{\tau}$	0.00450	1.28	0.00485	1.23
Without $\mathcal{L}_{\text{pred}}$	0.00475	1.28	0.00494	1.27
Without $\mathcal{L}_{\text{align}}$	0.00796	3.84	0.00976	3.11
Without $\mathcal{L}_{\text{desc}}$	0.0102	4.09	0.779	4.09
Only $\mathcal{L}_{\text{desc}}$	1.62	4.09	1.63	3.22

Table 1: Table of ablation study results. For each ablation we evaluate the MSE of trajectory decoder ($\mathcal{L}_{\text{pred}}$) as well as the cross entropy of the language description ($\mathcal{L}_{\text{desc}}$). We evalute these metrics on the validation set and show the best MSE throughout the training along with the CE at that iteration as well as the best CE and corresponding MSE at that iteration.

ing the language decoder $D_l(z_2 - z_1)$.

On the validation set, our model achieves a cross entropy of 1.15. Figure 4 compares this quantitative evaluation with a qualitative evaluation.

5 Ablation Study

To evaluate the effectiveness of our various losses, we conduct an ablation study on our model for the 2D navigation environment. As shown in Table 1, our complete model offers the best generalization on data in the validation set. It is interesting to note that removing the description loss $\mathcal{L}_{\text{desc}}$ results in worse generalization of the trajectory predictor as indicated by the MSE. This leads us to believe that learning a embedding space in which it is easier to infer language instructions between behavior points leads to a better structured, more generalizable space.

6 Discussion and Future Work

We proposed a model that learns a joint embedding space for behavior and language. We showed how the model can enable agents to adjust to human feedback and describe changes in behavior. It remains to scale this method to more complex tasks and language. Moreover, it would be exciting to see how a model trained using this method can be used to solve real world tasks using language feedback and offer explanations that shed light to often blackbox optimization methods.

7 Collaboration

This research project is a joint collaboration with Abhishek Gupta and Sergey Levine. I implemented the neural network models and losses, including the trajectory encoder/decoder and language encoder/decoder. I collected trajectories for reaching and navigation and created a dataset of labels. I created and executed all experiments, generated all figure, and wrote all parts of this report.

References

Riad Akrour, Marc Schoenauer, and Michele Sebag. 2011. *Preference-Based Policy Learning*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 12–27.

Riad Akrour, Marc Schoenauer, and Michèle Sebag. 2012. *APRIL: Active Preference Learning-Based Reinforcement Learning*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 116–131.

Riad Akrour, Marc Schoenauer, Michèle Sebag, and Jean-Christophe Souplet. 2014. Programming by Feedback. In *International Conference on Machine Learning*. JMLR.org, Pékin, China, number 32 in JMLR Proceedings, pages 1503–1511. https://hal.inria.fr/hal-00980839.

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*. PMLR, International Convention Centre, Sydney, Australia, volume 70 of *Proceedings of Machine Learning Research*, pages 166–175. http://proceedings.mlr.press/v70/andreas17a.html.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1:49–62. https://transacl.org/ojs/index.php/tacl/article/view/27.

Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Lawson L. S. Wong, and Stefanie Tellex. 2017. Accurately and efficiently interpreting human-robot instructions of varying granularities. *CoRR* abs/1704.06616. http://arxiv.org/abs/1704.06616.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* .

S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pages 82–90. http://dl.acm.org/citation.cfm?id=1687878.1687892.

SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 126–135.

Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741* .

Christian Daniel, Oliver Kroemer, Malte Viering, Jan Metz, and Jan Peters. 2015. Active reward learning with a novel acquisition function. *Auton. Robots* 39(3):389–405. https://doi.org/10.1007/s10514-015-9454-z.

Michaela Jänner, Karthik Narasimhan, and Regina Barzilay. 2017. Representation learning for grounded spatial reasoning. *CoRR* abs/1707.03938. http://arxiv.org/abs/1707.03938.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael L. Littman, Smaranda Muresan, Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. 2015. Grounding english commands to reward functions. In *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*. http://www.roboticsproceedings.org/rss11/p18.html.

James MacGlashan, Mark K. Ho, Robert Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E. Taylor, and Michael L. Littman. 2017. Interactive learning from policy-dependent human feedback. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*. PMLR, International Convention Centre, Sydney, Australia, volume 70 of *Proceedings of Machine Learning Research*, pages 2285–2294. http://proceedings.mlr.press/v70/macglashan17a.html.

James MacGlashan, Michael Littman, Robert Loftin, Bei Peng, David Roberts, and Matthew Taylor. 2014. Training an agent to ground commands with reward and punishment.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. Phoenix, AZ, pages 2772–2778.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Dipendra Kumar Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. *CoRR* abs/1704.08795. http://arxiv.org/abs/1704.08795.

Lanbo She and Joyce Chai. 2017. Interactive learning of grounded verb semantics towards human-robot communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1634–1644.

G. Suddrey, C. Lehnert, M. Eich, F. Maire, and J. Roberts. 2017. Teaching robots generalizable hierarchical tasks through natural language instruction. *IEEE Robotics and Automation Letters* 2(1):201–208. https://doi.org/10.1109/LRA.2016.2588584.

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning language games through interaction. *CoRR* abs/1606.02447. http://arxiv.org/abs/1606.02447.

Ziyu Wang, Josh Merel, Scott Reed, Greg Wayne, Nando de Freitas, and Nicolas Heess. 2017. Robust imitation of diverse behaviors. *arXiv preprint arXiv:1707.02747* .