Calibrated Feedback for Reinforcement Learning

Clara Cannon
Department of Computer Science
University of Texas at Austin
claracannon@utexas.edu

Abrar Anwar
Department of Computer Science
University of Texas at Austin
abraranwar@utexas.edu

Abstract

In this paper, we introduce a new concept to calibrate Reinforcement Learning (RL) called "supervised attention" from human feedback which focuses on novel task learning from human interaction on relevant features of the environment, which we hypothesize will allow for better model calibration from limited training data. We wanted to answer the following question: does model calibration improve the performance of an agent in reward-sparse environments given language feedback? We show that language helps the model predict actions with more confidence. We tested many methods for implementing this concept and settled on incorporating language feedback via a template matching scheme. Using deep learning methods, we translate human linguistic narration to a saliency map over the perceptual field. This saliency map is used to inform a deep-reinforcement learning system where features in the visual observation are most important relative to its position in the environment and optimize task learning. We establish a baseline model using deep TAMER with a replay buffer and test our framework on Montezuma's Revenge, the most difficult game in the Atari Arcade suite. Our approach uses feedback-based rewards to calibrate the uncertainty estimates for each action, and uses this estimate to improve exploration. Our results show that calibrated uncertainty estimates allows the agent to improve significantly over the other agents when using language-based feedback.

1 Introduction

Advances in machine learning have contributed to impressive progress in the development of reinforcement learning agents' ability to produce robust policies that generalize across differing situations seen and unseen during training [3, 12]. However, these methods typically only work under carefully designed testing conditions or simulated environments where experts hand select features [1], reward functions, and initial conditions [19]. This scheme works great under the assumption that an RL expert is on hand to ensure the optimal training and testing of the intelligent agent occurs without mishap. However, if we truly wish to develop general purpose agents on a wide scale for real world tasks under the tutelage of a non-expert, a more sustainable model is needed to quickly and robustly characterize novel tasks in unknown environments, utilizing information more intuitive to the average person, rendering the intervention of expert trainers unnecessary.

In the quest to realize this ideal, various approaches have been employed to allow a human user to train a RL agent through our natural forms of instruction and interaction, the most relevant to our project being language. One of the first ground breaking approaches and the one we reference the most throughout the remainder of this paper is TAMER [10] and its extension into deep neural architectures, deep TAMER [23]. These are two methods for allowing a human to provide continuous evaluative feedback on an agent's performance. They demonstrate how scalar feedback can significantly decrease learning time. While most other areas of research in the Artificial Intelligence (AI) domain might require training episodes to the scale of thousands and hundred thousands, very rarely do we see a RL

Preprint. Under review.

training scheme less than the order of millions (and that is being conservative). Deep TAMER also addresses the credit assignment problem. By assigning each action sequence a window of relevancy for each received feedback signal, they were able to discern which aspects of the current state made a particular action a good or bad option. However, one must admit the scalar reward as feedback is rudimentary and more tailored to the digital world than our own. We want to show that language feedback is just as valuable if not more significant to shortening agent learning and improving the calibration of a model.

Why do we think language is going to be so valuable in the RL domain? First, consider how humans learn and teach each other new skills. When a person is providing feedback or demonstrating a task for another person, they can describe what they are doing in natural language, providing context, clarification, and/or explanations for their evaluations or actions. Therefore, this work focuses on enabling intelligent agents to perform efficient and robust learning from feedback by leveraging auxiliary natural language narration as context. We show that this contextual information allows agents to intelligently disambiguate, generalize, and rapidly learn from human instruction a complex task in a sparse reward environment, the Atari game Montezuma's Revenge. We develop, implement, and evaluate our new approach to using language to aid task learning. We were inspired by ideas from language grounding, explanation for deep learning, and learning from rationales.

Our approach uses language narration as a supervisory signal that focuses learning on relevant features of the environment, thereby allowing effective learning from limited training data and enhancing model calibration. Our system pre-processes image inputs using language to focus agent's perception on task or environment relevant features. For instance, given Room 1 in Atari's Montezuma's Revenge, the human trainer can specify local optimal behaviours with language given the agent's progress towards the goal (key). When the agent is at the top of the ladder, we do not want the agent to immediately focus on moving left towards the key, as there is no viable path to reach the key's location from the starting position. Instead, a trainer might instruct the agent to move right, away from its primary objective, but along an optimal trajectory for the given environment. By leveraging prior work in video captioning [22, 21] and template matching, we construct a model and train it to use language embeddings within the environment observation space. The human linguistic narration is used to generate a saliency map over the perceptual field using a combination of methods described in [9] and [7].

Finally, this saliency map is passed to the deep reinforcement learning system from deep TAMER [23] with a replay buffer augmented to accept natural language feedback. We distinguish ourselves from previous work by [4] and [9] by focusing on natural language as feedback as opposed to a scalar-valued reward signal or set of instructions. Additionally, we model the uncertainty in the language feedback with respect to its observation using model calibration techniques. Language is incorporated solely as a supervised attention signal over the features of the high dimensional state observation.

This work covers the development, implementation, and experimental evaluation of our novel method for improving model calibration by augmenting agent learning with human feedback through the exploitation of information gathered from linguistic narration. We evaluate our approach on Montezuma's Revenge, the most challenging game in the Atari Arcade Suite. We tested our agent on Room 1 using a fixed reward function across all training and testing sessions. Overall, our project dives into and investigates an entirely different use of language– evaluative narration of agent performance. Our new method has the potential to significantly expand the field of RL and but also draw attention to how far we still need to go before at-home personalized robot companions can actually be realized.

2 Background and Related Works

2.1 Background

We make the common assumption that our learning task can be represented as a Markov Decision Process specified by the tuple (S, A, T, γ, D, R). S and A are the sets of possible states and actions respectively. T is a transition function, $T : S \times A \times S \rightarrow R$, which gives the probability, given a state and an action, of transitioning to another state on the next time step. γ, the discount factor, exponentially decreases the value of a future reward. D is the distribution of start states. R is a

reward function, $R : S \times A \times S \rightarrow R$, where the reward is a function of the most recent state, the most recent action, and the next state s_t, a_t, and s_{t+1}.

Reinforcement learning algorithms [20] seek to learn policies ($\pi : S \rightarrow A$) for an MDP that maximize return from each state-action pair, where *return* $= \sum_{t=0}^{t} [\gamma^t R(s_t, a_t)]$. Within reinforcement learning, there are two general approaches to this problem. Policy-search algorithms fix the values of some set of parameters, observe the mean return received for the fixed policy over some number of episodes, and then use a rule to determine what parameter values to try next. The other reinforcement learning approach models the expected return, or value, of a state or state-action pair when following a certain policy. Usually, the action with the highest expected return is selected (though sometimes with exploratory actions instead), and the agent updates the expected returns based on its experience. We use the latter approach when formulating our problem.

2.2 Calibration

The two most common approaches to calibration are Platt scaling [15] and isotonic regression [14]. Calibration has recently been applied to problems in deep neural networks [8] and model-based reinforcement learning [13]. In model-based reinforcement learning, [13] developed a general calibration technique for reinforcement learning which probabilistically modeled the observations to actions. They use the calibrated uncertainties to improve exploration using upper confidence bounds (UCB). In this work, we use a similar approach to improve agent exploration using human feedback-based rewards.

2.3 Saliency Maps

Deep learning systems are often treated as black boxes, where the inner workings of the systems are esoteric and hard to interpret. This led to work in generating explanations that help interpret the decisions made by deep networks such as CNNs [5] and RNNs [16]. Systems such as Grad-CAM [17] and Caption-guided visual saliency [16] are able to analyze a network's processing of a particular example and generate a "heat map" showing which features of the input most influenced the generated output. Caption-guided visual saliency takes a video captioning neural network and a given input/output pair, and produces heat maps over the input frames denoting which parts most influenced the computed output. EXPAND (EXPlanation AugmeNted feeDback) by [7] uses a human in the loop RL framework to provide visual explanations from saliency maps and binary feedback. They showed that the addition of state salient information boosts agent performance. Along this vein, we used a technique called template matching in conjunction with the Gaussian perturbation over the pixels to produce heat maps from state observations. Template matching is a technique that identifies the parts on an image that match a predefined template. We generate these heat maps from natural language feedback and use them to supervise the training of our model weights.

2.4 TAMER and Deep TAMER

In our work, we concentrate on two of the most popular frameworks for learning from human feedback, TAMER [10], and Deep TAMER [23]. We augmented the Deep TAMER framework to use linguistic information collected from human feedback to produce saliency maps.

The TAMER Framework is an approach to the Shaping Problem [2]. TAMER assumes human reinforcement to be fully informative about the quality of an action given the current state. It uses established supervised learning techniques to model a hypothetical human reinforcement function, $H : S \times A \rightarrow R$, treating the scalar human reinforcement value as a label for a state-action sample. Briefly, the TAMER algorithm operates in the following manner. First, the agent receives a scalar reward from the human trainer for the previous time step. If the human reward is nonzero, then the error is calculated as the difference between the human reward and model predicted reward, $\hat{H}(s, a)$. The loss is then propagated backward, along with the previous feature vector, to update the model. In the forward pass, when the agent takes a single step in the environment to reach a new state, it selects the action with the largest predicted reward according to the model.

Deep TAMER, an extension of the TAMER framework, leverages the representational power of deep neural networks in order to learn complex tasks in just a short amount of time with a human trainer.

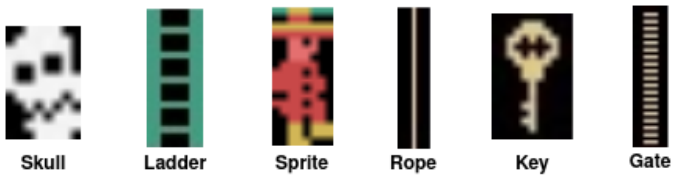

Figure 1: This figure depicts the templates we extracted from Room 1 of Montezuma's Revenge.

We use this approach to implement our supervised attention model with human feedback instead of a binary reward input from the human trainer.

3 Language Guided Template Matching

Our approach uses language narration to supervise the template matching saliency map generation. We hypothesise that instead of allowing the model to have access to all image features for every step in the training episode, thereby having to figure out on its own which features are the most important predicting reward and which ones can be ignore through a series of trials and errors, calibration will be greatly improved if we simply hand over the features of the image we think are most important. Having a selection of the features is useful for efficiency but designing them by hand is not scalable – therefore we use language. We test this by transferring the language information obtained from the human trainer to the image observation and outputting a heat map. The saliency map is supposed to help focus model weights on relevant features of the environment, thereby allowing more effective learning and a more calibrated model. We follow the language template matching scheme outlined in [9] with free form natural language feedback to generate a mask. We then apply a Gaussian filter over the original image to perturb the pixels and combine this blurred observation to obtain the final saliency map [6]. Applying Gaussian perturbations over irrelevant regions adds spatial uncertainty and motivates the agent to focus more on the clear relevant regions. We experimented with other perturbation methods, but found the Gaussian filter with blur radius of 3 worked the best for our purposes.

3.1 Using Language to Produce Saliency Maps

A fundamental challenge in RL is given a reward for a particular sequence of states and actions, ascertaining which features of the states and actions are most important for determining the reward. Using language to describe why an agent's actions were good or bad is valuable information. Our approach allows a neural model to focus on relevant state features communicated via a saliency map (aka heat map) highlighting the importance of image features through pixel weighting or masking.

We first implemented an algorithm that maps the language to the perception of the environment using templates we created (See Figure 1). Formally, given a natural language feedback provided from a synthetic language generator, we parse the utterance into separate word tokens. With these tokens, we preform a search across the generated template for the Montezuma's Revenge environment. The templates are tightly bounded images of important obstacles and features of the environment, including ladders, gates, the agent, the skull, and the key. We select the templates by matching word tokens in the feedback language to the template name. Once all of the relevant generated templates are gathered based off of the language feedback, we compute a mask over the observation features. The mask leaves the important features in the observation space and masks out all others. We then apply a Gaussian filter to the original image which consequently blurs pixel appearance. After applying the mask to the now blurry observation, the resulting array contains the features with relevant object information with unimportant features obscured (see Figure 2). In essence we generate heat maps over the input frames denoting which objects most influenced the computed output. This heat map is then passed as an input during training to deep RL model in a supervised setting, allowing the system to use the language to improve learning from feedback.

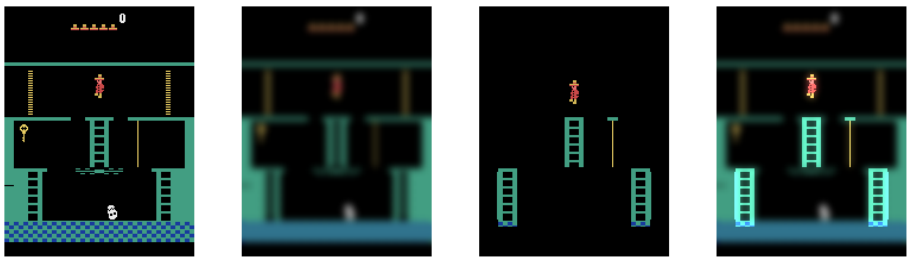

Figure 2: Saliency Map Generation. We produce heat maps for the model based on a pseudo random interval to authentically simulate a human in the training loop. A Gaussian filter with blur radius of 3 is applied to the observation image. The resulting vector is added to the template mask. The final saliency map is depicted in the far right image of the figure.

3.2 Template Matching in Atari

We use the cv2 library in Python to perform the template matching. Template matching is a method for searching and finding the location of a template image in a larger image. The OpenCV library comes with a built in that slides the template image over the input image and compares the template and patch of input image under the template image. It returns a grayscale image, where each pixel denotes how much the neighbours of that pixel match with the template. Once we have the result, we can find the maximum/minimum value. However, this will not provide us will all the locations of an object when multiple occurrences of the same template are found.

For our purposes, we implemented template matching with multiple objects. Consider Room 1 in Montezuma's Revenge. If the human trainer provides feedback such as "Climbing down the ladder was a good action", we want to locate all of the ladders across the observation so the model can interpret that ladders should be associated with climbing up and down actions. In this case, we used a thresholding approach. If a pixel value is greater than the threshold value, it is assigned 1 else it is assigned 0. In our case, the threshold is 1, meaning the pixel in the template is a direct match for the pixel in the observation image. Next we compare a template against overlapped image regions. As we slides through sections of the image, we compare the overlapped patches of size $w \times h$ against the template and store the comparison results. We used the template matching normalized correlation coefficient comparison method:

$$\hat{I}(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y')) \sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2} \quad (1)$$

where I denotes image, T template, $\hat{I}$ result. After the function finishes the comparison, the best matches are found as global maximums.

4 Deep TAMER + *Feedback*

Deep TAMER [23] extends TAMER [10] a LfD framework. A human observes an autonomous agent trying to perform a task in a high-dimensional environment and provides scalar-valued feedback as a means by which to shape agent behavior. Deep TAMER + *Feedback* enriches the Deep TAMER observation space with natural language feedback, potentially enabling better sample efficiency and training time.

4.1 Algorithm

We use the problem formulation proposed in [23] with a few modifications . Instead of a binary reward signal, we use language feedback collected from human trainers. We also modify the binary reward signal in order to standardize non-environment reward during training. More formally, Let S denote the set of states in the agent's environment, and let A denote the set of legal actions the agent can execute. The execution of actions $(a_1, a_2, \cdots, a_n)$ result in a state trajectory $\tau = (s_0, s_1, s_2, \cdots, s_n)$

where n is either the terminal state in an episode of the training horizon ($<$ the max number of training steps). The human trainer observes the state trajectory and provides natural language feedback, $(\ell_1, \ell_2, \cdots)$, that convey their personal evaluation of the agent's behavior. We hard code the scalar-valued feedback signals, $(h_1, h_2, \cdots)$, as a means to optimize the training process and allow for uniformity across training sessions. We hypothesize that improved agent performance would be the result of quality language feedback, not a dense reward function. We model the reward signal $R(s, a)$ for a given state and action pair by taking the Euclidean distance d between the agent and nearest critical point along the optimal trajectory, determined by a human expert (see Figure 3).

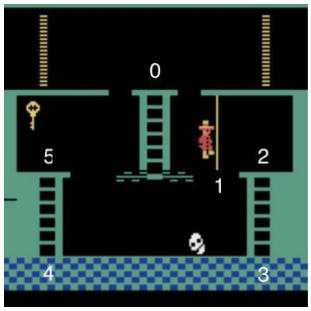

Figure 3: Map of the critical positions encoded in Room 1 of Montezuma's Revenge. These positions inform the reward function $R(s, a)$ the appropriate scalar reward signal to feed to the environment.

$$d = \sqrt{(x - x')^2 + (y - y')^2} \tag{2}$$

$$R(s, a) = \{\, 1 \; d < \alpha 0 \alpha < d < \beta - 1 \beta < d \tag{3}$$

where the lower threshold β and upper threshold α are set the 8 and 20 respectively. We determined the best threshold values from repeated trial and error (or hyperparameter search). We also penalize the agent with negative reward if it is stationary in the same state for more than 10 time steps. We do this because early on in training, we noticed the agent tends to hide in the regions of the environment with 0 reward. Thus to encourage exploration and prevent the agent from getting "stuck", we include this extra reward shaping condition.

When ℓ_i is available, we use it to compute the saliency map using the template matching technique described in Chapter 3. Then, we compute the predicted reward given the original state observation s,and the predicted reward with the saliency map s'. When the model is updated and the loss propagated backward through the weights, we use these two rewards to compute an $L2$ regularization and add it onto the calculated MSE loss. We include this additional loss to force the model weights to 0 for unimportant state observation features.

$$L2 = \sum_{i=1}^{n} |y_{original} - y_{saliency}|^2 \tag{4}$$

where $y_{original}$ is the predicted reward when the original observation is fed as input into our deep model and $y_{saliency}$ is the predicted reward given the saliency map as input. When ℓ is not available, we ignore the L2 loss entirely and set it to 0. Now, our completed loss function that we aim to minimize looks like:

$$L = MSE + \alpha \cdot L2 \tag{5}$$

where the regularization rate is 0.0001. We adopt a greedy policy for action selection. Given the vector of predicted rewards from the model, the agent selects the action associated with the greatest reward.

4.2 Calibration

For calibration, we develop an approach similar to [13]. Since uncertainty estimates are required, we use a Bayesian neural network (BNN) to provide uncertainty estimates to each action. This provides us with a mean and variance estimate of the expected human-based reward for each action. As these

variances are not calibrated with respect to the frequency of these actions, we apply Platt scaling to recalibrate the network at the end of each episode. This is done by sampling from the replay buffer, freezing all weights in the network except for the parameters of the Platt scaling, and training for a number of steps. During typical training, we freeze the Platt scaling parameters.

These uncertainty estimates can then be used with a UCB-style algorithm to choose the action which has the highest reward confidence interval. In particular, we use UCB to derive an action:

$$a_t^{UCB} = argmax_{a \in \mathcal{A}} \hat{Q}(A) + \hat{U}(a). \tag{6}$$

Our calibrated agent is then takes advantage of the saliency map and the natural language feedback to produce potentially high reward behaviors through exploration.

5 Experiments and Results

We pre-trained an auto encoder for feature extraction on the image data set from [4], which was adapted from the Atari Grand Challenge data set [11], which contains hundreds of crowd-sourced trajectories of human game plays on 5 Atari games, including Montezuma's Revenge. We only used image from Room 1 in Montezuma's Revenge, which amounted to roughly $2,000$ images with which to train our auto encoder.

We used the Deep TAMER model with a replay buffer, which we will henceforth refer to as TAMER Repay, as our baseline. While the Deep TAMER with Credit Assignment was the best performing model in [23], the success was largely based on the hand selected credit assignment window. We chose a simpler TAMER implementation to narrow the field of experimental variables in order to better highlight the effects language had on model calibration. We focus more on exploring the benefit of language throughout the training process rather than task completion. We set the training interval to $10,000$ steps, with 100 warm up steps at the beginning of each trial to introduce variety in the agent starting position. We compute performance as an average across 10 sessions. This was necessary due to the extreme variability in training sessions. We could not rely on a single session to demonstrate an authentic data trend.

Current Position	Feedback
-1	"Go back to the top of the ladder and jump onto the rope."
0	"Jump right from the ladder onto the rope."
1	"Jump right from the rope onto the lower platform."
2	"Climb down the ladder onto the ground."
3	"Move left and avoid the skull."
4	"Climb up the ladder."
5	"Jump up to get the key."

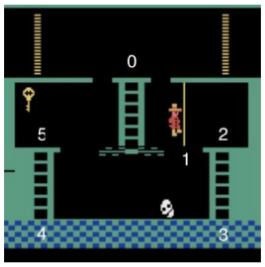

Figure 4: Synthetic Feedback Language. Table showing the synthetic language assigned to each critical position along the optimal trajectory. The position -1 is everywhere else in the environment that is not marked with a position numbered $0-5$. Each utterance contains the template name that we reason are important for the model to focus in that stage of the task.

Our initial intention was to collect natural language feedback, however we settled on using synthetic language due to time constraints. The language feedback was hard coded into a generator function that returned the most applicable feedback given the agent's current position and its previous position (See Figure 4). As mentioned previously, we captured the agent's progress throughout Room 1 in Montezuma's Revenge by marking "critical positions" along the optimal trajectory set by a human expert. When the agent found itself in a critical position, the language generation function would deliver feedback aimed toward getting the agent to the next critical position. When the agent is not on the optimal path or in transition from one critical position to the next, its position is labeled -1. When deciding which feedback is best suited for transitionary states (when the agent's current position is -1), we need to consider its position along the optimal path. We find the agent's distance to the nearest critical position, all lives intact, and send feedback relevant to reaching that location.

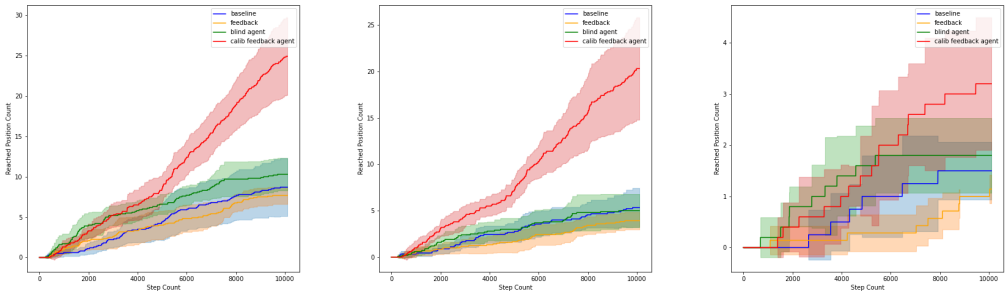

Figure 5: These results show the mean number of times the agent reached each position 1, 2, and 3 over 10 trials along with confidence bounds. (Left) The number of times each agent reached position 0. (Middle) The number of times each agent reached position 1. (Right) The number of times each agent reached position 2.

Because the saliency map is being generated via template matching scheme, the actual semantics of the feedback is less important than the content from which we extract tokens used to search the template space for matches. The agent often gets "stuck" in the upper corners of environment near the gates. In this case, we use a position buffer to track its movement. If the position buffer contains the same coordinates, we set the appropriate flag and stop feeding the model saliency maps. The intuition behind this decision is that the agent will become unstuck faster when the model receives complete information of the environment. Since we do not use live human trainers for feedback, we also needed to reason about how often feedback would be available, and consequently saliency maps. It is unrealistic to expect a human to provided natural language feedback at every time step. Not only would this precedent be laborious for the participant, but contradictory to our goal of making agent training faster and more efficient. We settled on a feedback interval of 10 seconds after several trials, determining the parameter value struck the right balance between often enough and too much.

In Figure 5, we compute number of times the agent reaches a critical position along the optimal trajectory. We show the results for four agent-types: a baseline agent which does not use any feedback, a feedback agent that uses only the saliency map, a blind agent which uses natural language feedback with a blurred observation, and a calibrated agent which calibrates the blind agent. Our results show that the calibrated agent outperforms every other agent and learns faster. Additionally, all other agents never reached the fourth position; however, the calibrated agent reached that position twice on average at the end of training. Without calibration, language provides little increase in performance in Montezuma's revenge; however, a calibrated model allows the agent to make better decisions.

6 Conclusion

We explore a novel approach to Reinforcement Learning (RL) that uses natural language feedback to provide supervised attention. We compared a TAMER model with the additional of natural language generated saliency maps to the baseline and found language had no significant improvement or worsened performance on Montezuma's Revenge, a popular video game in the Atari Arcade suite. This led to the construction of a blind agent training scheme, where the salient information passed to the model only contained the agent position, which also showed little increase in performance. Language on its own was unable to allow the agent to proceed; however, a calibrated deep reinforcement learning model allows the agent to build a richer understanding of the environment and make better decisions. We conclude that model calibration of deep reinforcement learning allows the agent to take advantage of its inputs, such as language, and allows language-guided agents to learn faster in this reward-sparse environment.

More rigorous studies need to be conducted in this area to ensure the effectiveness of model calibration. Having uncertainty estimates may allow for interesting behaviors to emerge in tasks such as action advising [18], where the agent could ask for help based on a calibrated uncertainty estimate of its actions. These results can be leveraged in the construction of sample-efficient, feedback-driven reinforcement learning agents.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] M. E. Bouton. *Learning and behavior: A contemporary synthesis.* Sinauer Associates, 2007.

[3] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

[4] P. Goyal, S. Niekum, and R. J. Mooney. Using natural language for reward shaping in reinforcement learning, 2019.

[5] Y. Goyal, A. Mohapatra, D. Parikh, and D. Batra. Towards transparent ai systems: Interpreting visual question answering models. *arXiv preprint arXiv:1608.08974*, 2016.

[6] S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801. PMLR, 2018.

[7] L. Guan, M. Verma, S. Guo, R. Zhang, and S. Kambhampati. E, 2020.

[8] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks.

[9] R. Kaplan, C. Sauer, and A. Sosa. Beating atari with natural language guided reinforcement learning, 2017.

[10] W. B. Knox and P. Stone. Tamer: Training an agent manually via evaluative reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pages 292–297. IEEE, 2008.

[11] V. Kurin, S. Nowozin, K. Hofmann, L. Beyer, and B. Leibe. The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*, 2017.

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[13] A. Malik, V. Kuleshov, J. Song, D. Nemer, H. Seymour, and S. Ermon. Calibrated Model-Based Deep Reinforcement Learning.

[14] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning - ICML '05*, pages 625–632. ACM Press.

[15] J. C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

[16] V. Ramanishka, A. Das, J. Zhang, and K. Saenko. Top-down visual saliency guided by captions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7206–7215, 2017.

[17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[18] F. L. D. Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor. Uncertainty-Aware Action Advising for Deep Reinforcement Learning Agents. 34(04):5792–5799.

[19] D. Skinner. Mark e. bouton learning and behavior: A contemporary synthesis. *CANADIAN PSYCHOLOGY*, 48(4):281, 2007.

[20] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, 2011.

[21] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*, pages 4534–4542, 2015.

[22] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

[23] G. Warnell, N. Waytowich, V. Lawhern, and P. Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.