Learning with Latent Language

Jacob Andreas, Dan Klein, Sergey Levine

CS330 Student Presentation

Motivation

The structure of natural language reflects the structure of the world.

The authors propose to use language as a latent parameter space for few-shot learning.

Experiment with tasks including classification, transduction and policy search.

They aim to show that this linguistic parameterization produces models that are both more accurate and more interpretable than direct approaches to few-shot learning

Method

Methods for training is 2 fold:

1. Encoder - Decoder model for learning language representations
2. Classic few shot meta learning models

Authors import the relevant structure for problem solving from the first stage and utilize that in the second.

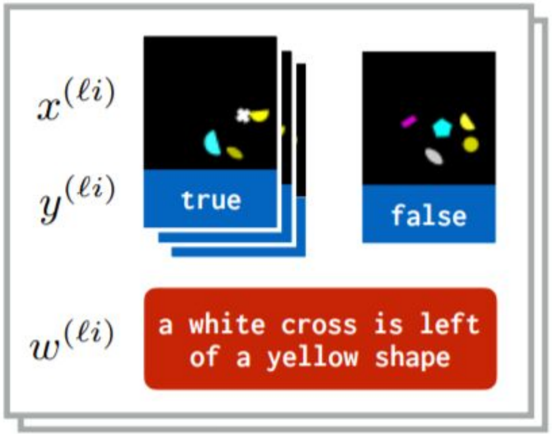
They achieve this in 3 steps:

1. Model Pre-Training / Language Learning Phase
2. Concept Learning Phase
3. Evaluation Phase

Method - PreTraining/ Language Learning

1. Involves pre-training a language model on specific subtasks using natural language parameters *“w”*
2. A language interpretation model is also learned to turn a description w into a function from inputs to outputs.
3. These natural language parameters are only observed at language-learning time.

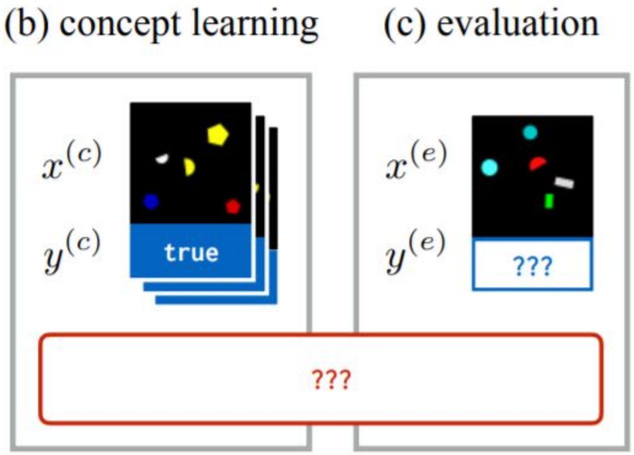
(a) language learning



$$\underset{\eta \in \mathbb{R}^a}{\arg\min} \sum_{i,j} L\big(f(x_j^{(\ell i)};\, \eta,\, w^{(\ell i)}),\, y_j^{(\ell i)}\big)$$

Method - Concept Learning

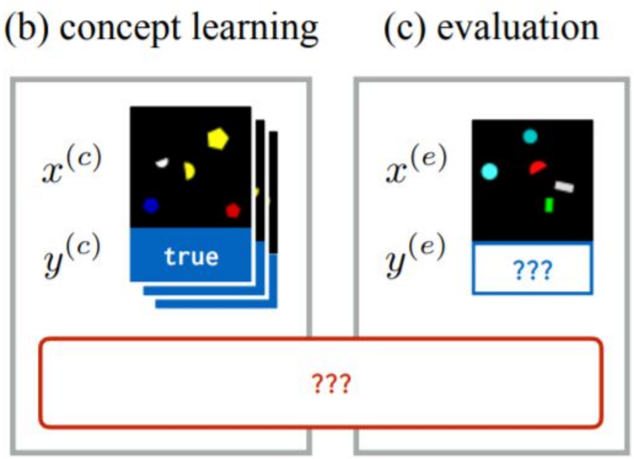
1. The pretrained model is adapted to fit data for a specific new task
2. Model generates natural language strings '*w^c*'
3. These are sampled from the model as approximations to the distribution of descriptions, given the task data.
4. By sampling from the pre-trained model, candidate descriptions are likely to obtain small loss.



$$\arg\max_\lambda \log q(w_i | x_1^{(\ell i)}, y_1^{(\ell i)}, \ldots, x_n^{(\ell i)}, y_n^{(\ell i)}; \lambda)$$

Method - Evaluation

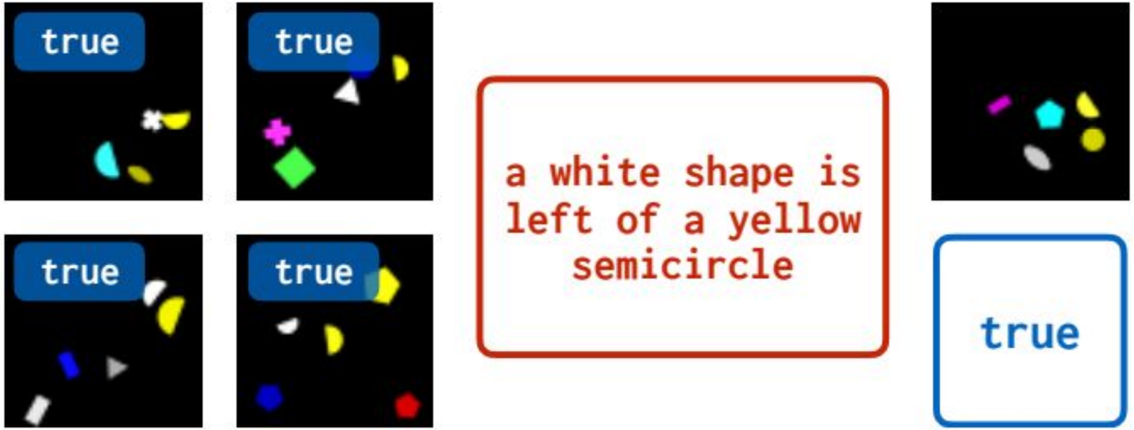
1. At evaluation time the hypothesis '*w^c*' that hopefully obtains the lowest loss is selected, and applied to a new task, i.e new input x to predict y.



Experiments

1. Few-Shot image classification
2. Programming by demonstration
3. Policy search

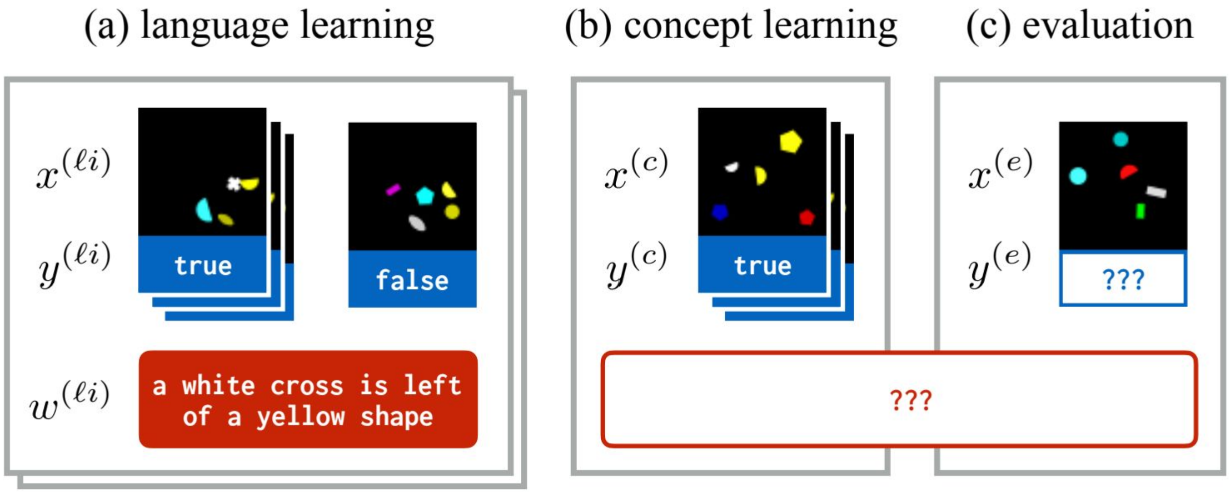
Experiments: few-shot image classification



Experiments: few-shot image classification

$$f(x; w) = \sigma\left(\texttt{rnn-encode}(w)^\top \texttt{rep}(x)\right)$$
$$q(w \mid \{x_j\}) = \texttt{rnn-decode}\left(w \mid \tfrac{1}{n} \textstyle\sum_j \texttt{rep}(x_j)\right)$$

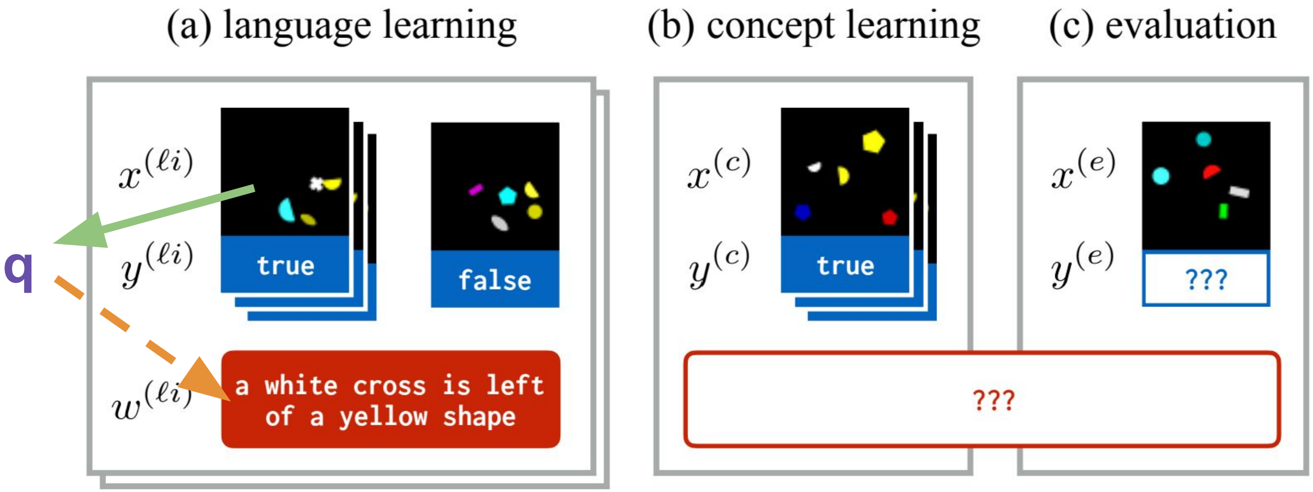
f performs task conditioned on task representation
q generates task representation as English sentence



Experiments: few-shot image classification

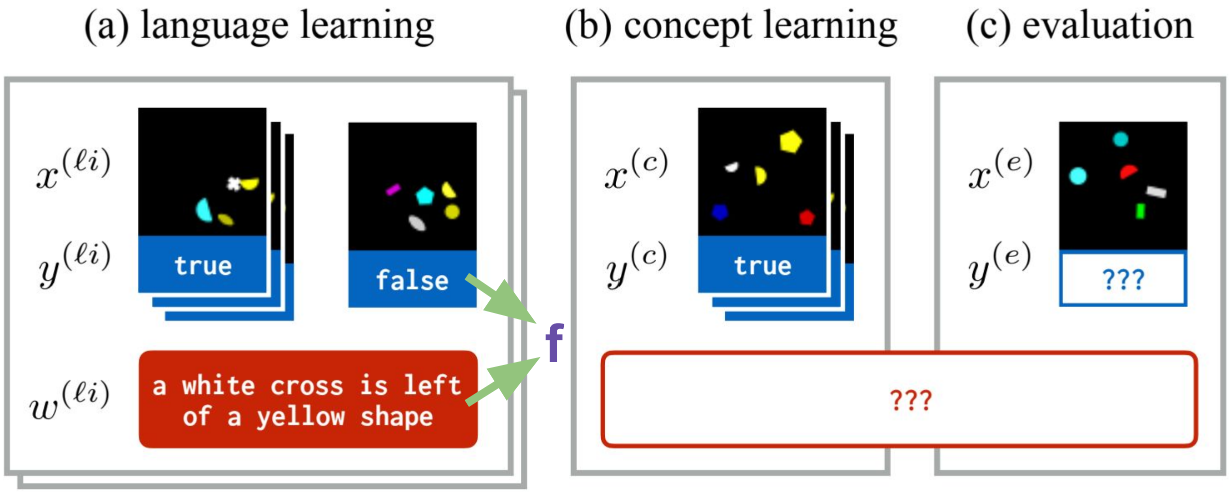
$$f(x; w) = \sigma\left(\text{rnn-encode}(w)^\top \text{rep}(x)\right)$$

$$q(w \mid \{x_j\}) = \text{rnn-decode}\left(w \mid \tfrac{1}{n} \textstyle\sum_j \text{rep}(x_j)\right)$$



Experiments: few-shot image classification

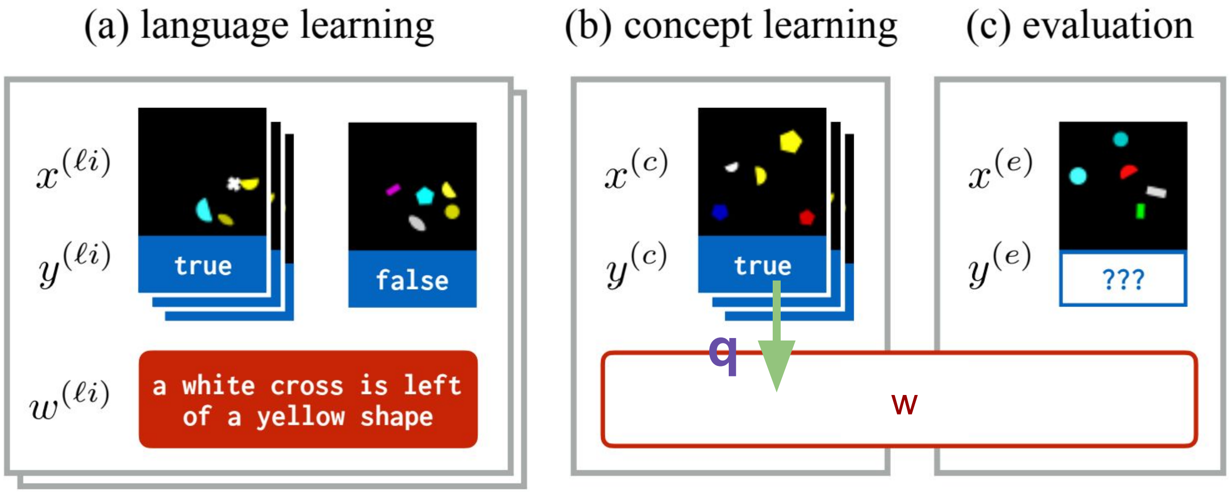
$$f(x; w) = \sigma\big(\textsf{rnn-encode}(w)^\top \textsf{rep}(x)\big)$$
$$q(w \mid \{x_j\}) = \textsf{rnn-decode}\big(w \mid \tfrac{1}{n} \textstyle\sum_j \textsf{rep}(x_j)\big)$$



Experiments: few-shot image classification

$$f(x; w) = \sigma\left(\texttt{rnn-encode}(w)^\top \texttt{rep}(x)\right)$$

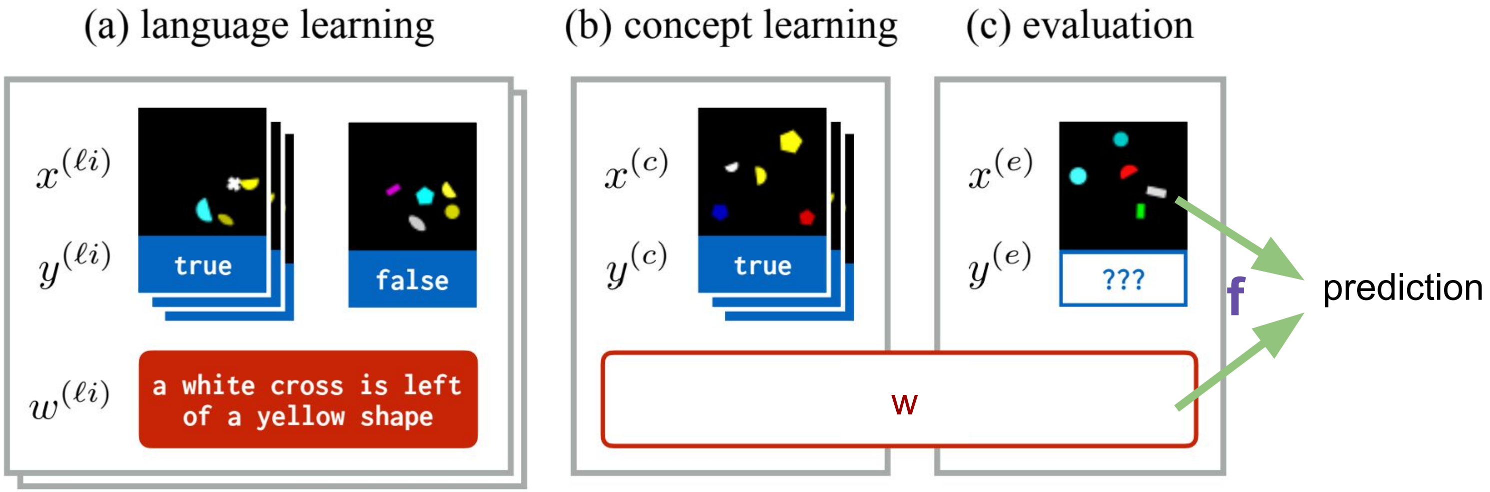
$$q(w \mid \{x_j\}) = \texttt{rnn-decode}\left(w \mid \tfrac{1}{n}\textstyle\sum_j \texttt{rep}(x_j)\right)$$



Experiments: few-shot image classification

$$f(x; w) = \sigma\left(\texttt{rnn-encode}(w)^\top \texttt{rep}(x)\right)$$

$$q(w \mid \{x_j\}) = \texttt{rnn-decode}\left(w \mid \tfrac{1}{n} \textstyle\sum_j \texttt{rep}(x_j)\right)$$



Experiments: few-shot image classification

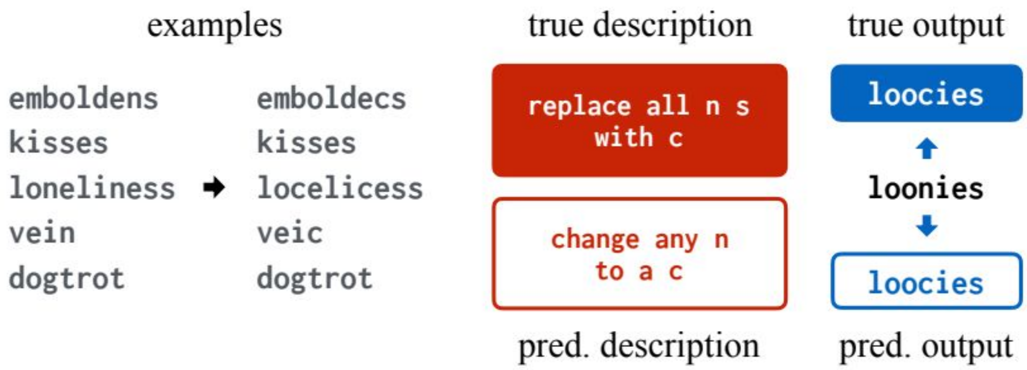
Model	Val (old)	Val (new)	Val	Test
Random	50	50	50	50
Multitask	64	49	57	59
Meta	63	62	62	64
Meta+Joint	63	69	66	64
L^3 (ours)	**70**	**72**	**71**	**70**
L^3 (oracle)	77	80	79	78

Experiments: programming by demonstration

$$\texttt{rep}(x, y) = \texttt{rnn-encode}([x, y])$$

$$f(y \mid x; w) =$$
$$\quad \texttt{rnn-decode}\big(y \mid [\texttt{rnn-encode}(x), \texttt{rnn-encode}(w)]\big)$$

$$q(w \mid \{(x_j,\, y_j)\}) =$$
$$\quad \texttt{rnn-decode}\big(w \mid \tfrac{1}{n} \textstyle\sum_j \texttt{rep}(x_j, y_j)\big)$$

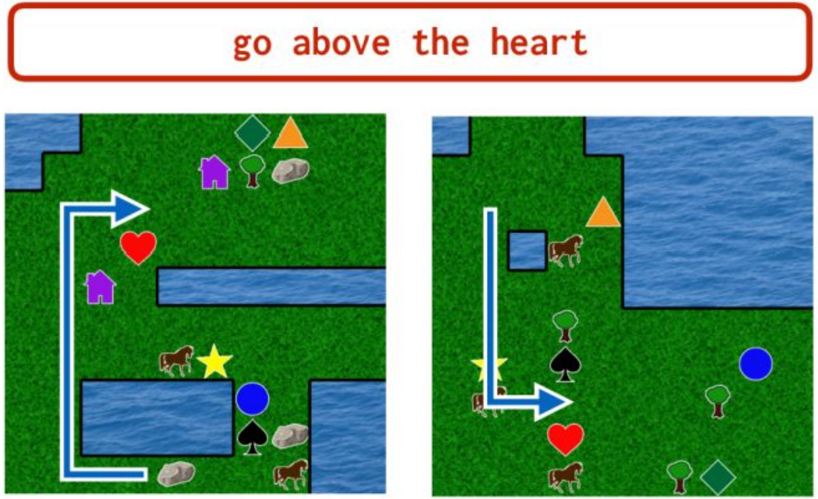


Experiments: programming by demonstration

Model	Val	Test
Identity	18	18
Multitask	54	50
Meta	66	62
Meta+Joint	63	59
L^3	**80**	**76**

Experiments: policy search

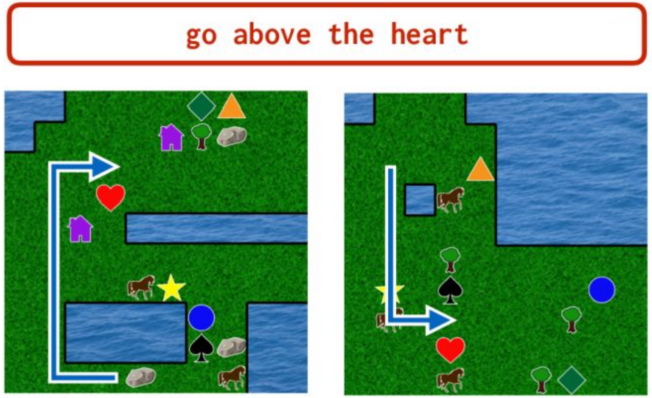
- Use latent language for structured exploration
- Imitation learning with expert trajectories



Experiments: policy search

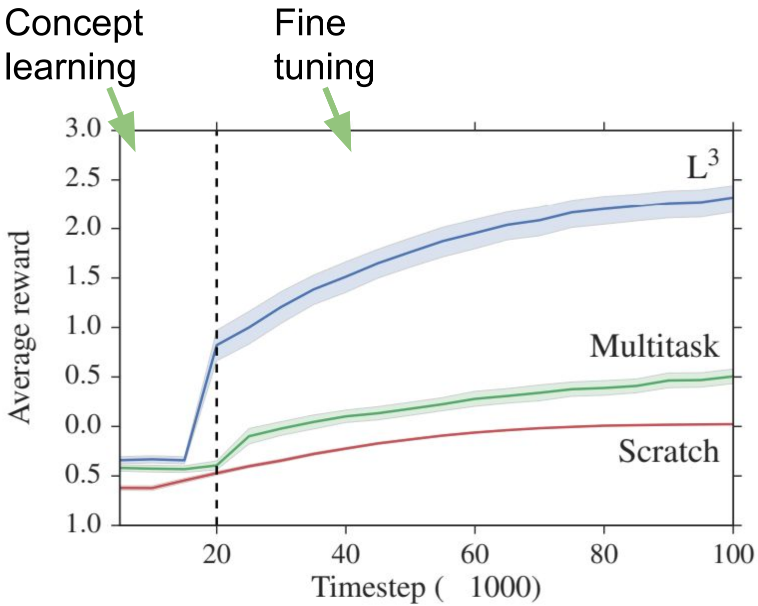
- Unconditioned **q**!

$$f(a \mid x;\, w) \propto \texttt{rnn-encode}(w)^\top W_a\, \texttt{rep}(x)$$
$$q(w) = \texttt{rnn-decode}(w)$$



Experiments: policy search

- Concept learning:
    - sample w from q to get exploration strategies
    - roll out policies conditioned on w
- Fine tuning:
    - policy gradient on best policy found in concept learning



Takeaways

Present an approach for optimizing models by using natural language as latent representations.

Approach outperformed some baselines on classification, structured prediction and reinforcement learning tasks.

Language encourages/allows for better compositional generalization - Few Shot

Language helps simplify structured exploration - RL

Discussion / Strengths and Weaknesses

- Not really clear what distinction is between concept learning and evaluation
- Good baselines for backing up their “philosophical” goal
- Limitation: need task-specific human language annotations
    - Challenge to move beyond toy examples
- Could this method be streamlined with an end-to-end approach?
    - Take cues from SeqGAN?