Learning with latent language

Structure

1. Introduction
2. Background
3. Learning With Language
4. Few-shot Classification
5. Programming by Demonstration
6. Policy Search
7. Conclusion

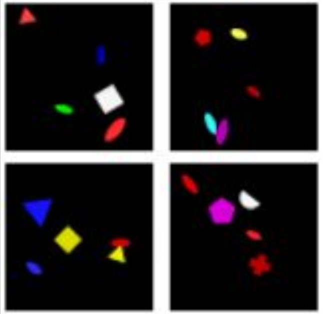
Introduction

- Can background knowledge from language improve the generality and efficiency of learned models?
- They present a model that uses the space of natural language strings as a parameter space to capture natural task structure
- Not require language data to learn new concepts: language is used only in pretraining

C Examples: ShapeWorld

(Examples in this and the following appendices were not cherry-picked.)

Positive examples:



True description:
a red ellipse is to the right of an ellipse

Inferred description:
a red shape is to the right of a red semicircle

Input:

True label:
true

Pred. label:
true

Few-shot classification

D Examples: Regular Expressions

Example in:	Example out:	Human description:	Input:	True out:
mediaeval	ilediaeval	leading consonant si replaced with i l	chaser	ilhaser
paneling	ilaneling			
wafer	ilafer	**Inferred description:**		**Pred. out:**
conventions	ilonventions	first consonant of a word is replaced with i l		ilhaser
handsprings	ilandsprings			

E Examples: Navigation

White breadcrumbs show the path taken by the agent.

Human description:
`move to the star`

Inferred description:
`reach the star cell`



`reach square one right of triangle`

`reach cell to the right of the triangle`



Introduction

New concepts are learned by searching directly in the space of natural language strings to minimize the loss incurred by the interpretation model.

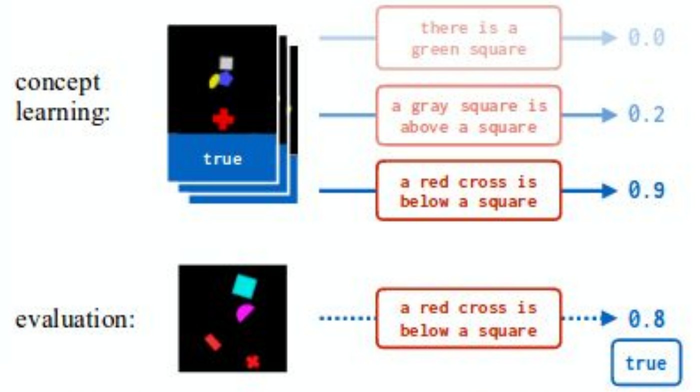


Figure 1: Example of our approach on binary image classification. In meta-training (not shown), we learn a language interpretation model that outputs the probability that an image matches a description. To learn a new visual concept, we optimize over descriptions to maximize the interpretation model's score. The chosen description can be used to classify new images.

Background: program synthesis

- Reduce the effective size of the parameter space **H** by moving the optimization problem out of the continuous space of weight vectors and into a discrete space of formal program descriptors
- They are also limited in their application: a human designer must hand-engineer the computational primitives necessary to compactly describe every learnable hypothesis.

Background: multitask learning

1. a **pretraining** (or "meta-training") phase that makes use of various different datasets i with examples $\{(x_1^{(\ell i)}, y_1^{(\ell i)}), \ldots, (x_n^{(\ell i)}, y_n^{(\ell i)})\}$ (Figure 2a)
2. a **concept-learning** phase in which the pretrained model is adapted to fit data $\{(x_1^{(c)}, y_1^{(c)}), \ldots, (x_n^{(c)}, y_n^{(c)})\}$ for a specific new task (Figure 2b)
3. an **evaluation** phase in which the learned concept is applied to a new input $x^{(e)}$ to predict $y^{(e)}$ (Figure 2c)

Pretraining:

$$\operatorname*{arg\,min}_{\eta \in \mathbb{R}^a,\, \theta^{(\ell i)} \in \mathbb{R}^b} \sum_{i,j} L\left(f\left(x_j^{(\ell i)};\, \eta,\, \theta^{(\ell i)}\right),\, y_j^{(\ell i)}\right)$$

Concept learning:

$$\operatorname*{arg\,min}_{\theta^{(c)} \in \mathbb{R}^b} \sum_{j} L\left(f\left(x_j^{(c)};\, \eta,\, \theta^{(c)}\right),\, y_j^{(c)}\right)$$

η - shared parameters, θ - task-specific parameters.

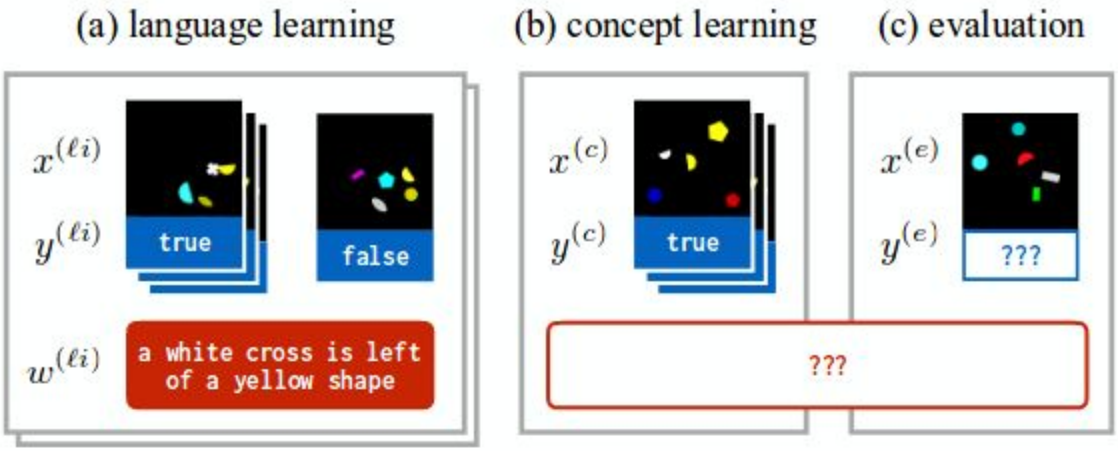


Figure 2: Formulation of the learning problem. Ultimately, we care about our model's ability to learn a concept from a small number of training examples (b) and successfully generalize it to held-out data (c). In this paper, concept learning is supported by a language learning phase (a) that makes use of natural language annotations on other learning problems. These annotations are not provided for the real target task in (b–c).

Learning with latent language

- At ***meta-learning*** time we additionally have access to natural-language descriptions $w^{(\ell i)}$
- Meta-learning == learning with language

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

- pretraining

concept learning -

$$\arg\min_{w' \in \Sigma^*} \sum_{j} L(f(x_j^{(c)}; \eta, w^{(c)}), y_j^{(c)})$$

Learning with latent language

In particular, use the language-learning datasets, consisting of pairs $(x_j^{(\ell i)}, y_j^{(\ell i)})$ and descriptions w_i, to fit a reverse **proposal** model, estimating:

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

q - provides a (suitably normalized) approximation to the distribution of descriptions given task data. (e.g. image captioning model)

By sampling from **q**, we expect to obtain candidate descriptions that are likely to obtain small loss.

We’ve got:

Generic procedure for equipping collections of related learning problems with a natural language hypothesis space

Next:

How this procedure can be turned into a **concrete algorithm** for supervised classification and sequence prediction, how to extend these techniques to reinforcement learning

Few-shot Classification

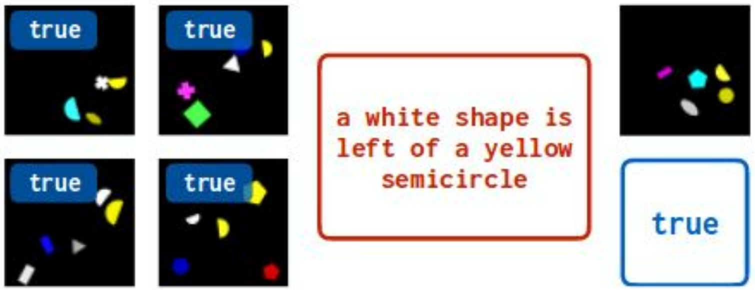
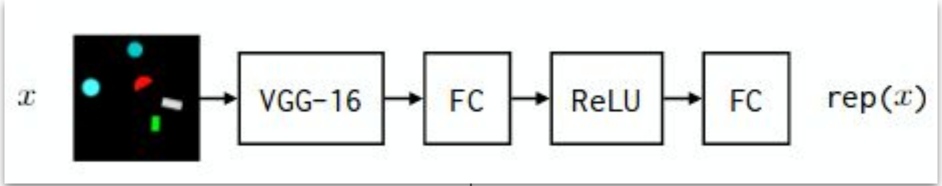


Figure 3: The few-shot image classification task. Learners are shown four positive examples of a visual concept (left) and must determine whether a fifth image matches the pattern (right). Natural language annotations are provided during language learning but must be inferred for concept learning.

Few-shot Classification: implementation



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

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

Baselines:

1. *Multitask*: a multitask baseline in which the definition of f above is replaced by $\sigma(\theta_i^\top \texttt{rep}(x))$ for task-specific parameters θ_i that are optimized during both pretraining and concept-learning.

2. *Meta*: a meta-learning baseline in which f is defined by $\sigma([\frac{1}{n}\sum_j \texttt{rep}(x_j)]^\top \texttt{rep}(x))$.[3]

3. *Meta+Joint*: as in *Meta*, but the pretraining objective includes an additional term for predicting q (discarded at concept-learning time).

Few-shot Classification: experiments

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

Table 1: Evaluation on image classification. *Val (old)* and *Val (new)* denote subsets of the validation set that contain only previously-used and novel visual concepts respectively. L^3 consistently outperforms alternative learning methods based on multitask learning, meta-learning, and meta-learning jointly trained to predict descriptions (*Meta+Joint*). The last row of the table shows results when the model is given a ground-truth concept description rather than having to infer it from examples.

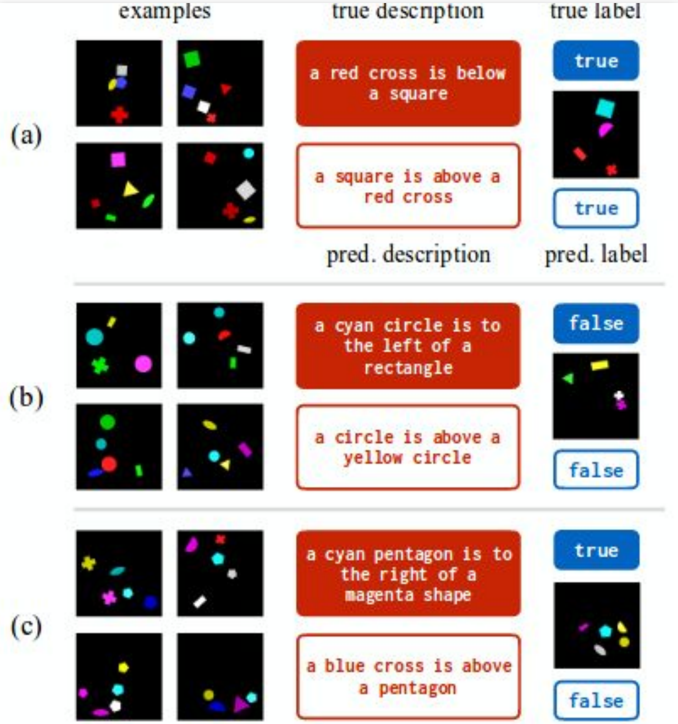


Figure 4: Example predictions for image classification. The model achieves high accuracy even though predicted descriptions rarely match the ground truth. High-level structure like the presence of certain shapes or spatial relations is consistently recovered. Best viewed in color.

results

Programming by Demonstration

Figure 5: Example string editing task. Learners are presented with five examples of strings transformed according to some rule (left), and must apply an appropriate transformation to a sixth string (right). Language-learning annotations (center) may take the form of either natural language descriptions or regular expressions.

Programming by Demonstration: implementations

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

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

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

Programming by Demonstration: experiments

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

Table 2: Results for string editing. The reported number is the percentage of cases in which the predicted string exactly matches the reference. L^3 is the best performing system; using language data for joint training rather than as a hypothesis space provides little benefit.

Annotations	Samples		Oracle	
	1	100	Ann.	Eval.
None (Meta)	*66*	–	–	–
Natural language	66	*80*	75	–
Regular expressions	60	76	88	90

Table 3: Inference and representation experiments for string editing. Italicized numbers correspond to entries in Table 2. Allowing the model to use multiple samples rather than the 1-best decoder output substantially improves performance. The full model does better with inferred natural language descriptions than either regular expressions or ground-truth natural language.

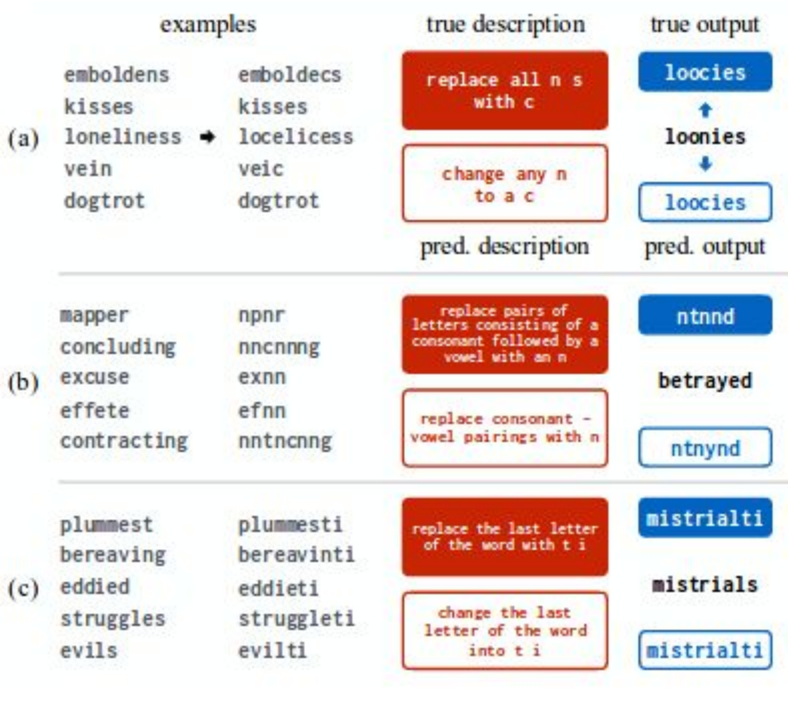


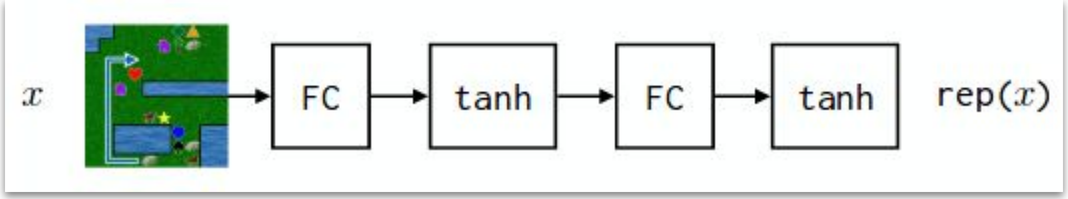
Figure 6: Example predictions for string editing.

results

Policy Search

- Goal here will be to build agents that can adapt quickly to new environments, rather than requiring them to immediately perform well on heldout data.
- Here the interpretation model **f** describes a policy that chooses actions conditioned on the current environment state and its linguistic parameterization.

Policy Search



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

$$q(w) = \text{rnn-decode}(w)$$

Policy Search: Training

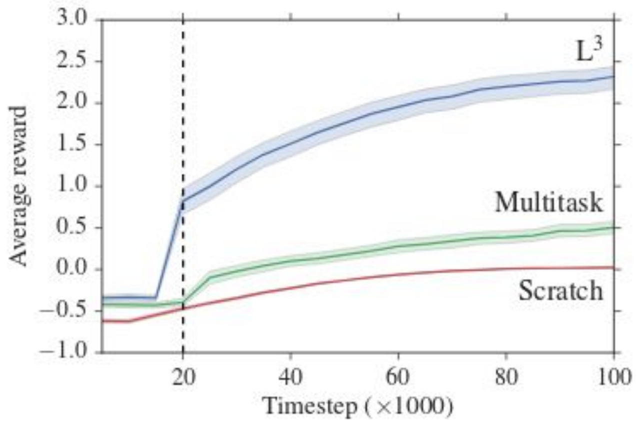


Figure 8: Learning curves for treasure hunting. These show the average reward obtained by each learning algorithm across multiple evaluation environments, after language learning has already taken place. *Multitask* learns a separate embedding for each task, while *Scratch* trains on every task individually. L^3 rapidly discovers high-scoring policies in most environments. The dashed line indicates the end of the concept-learning phase; subsequent performance comes from fine-tuning. The max possible reward for this task is 3 points. Error bands shows 95% confidence intervals for mean performance.

reach cell on left of triangle

reach square left of triangle

---

reach spade

go to the spade

---

left of the circle

go to the cell to the left of the circle

---

reach cell below the circle

reach cell below circle

Conclusions

- An approach for optimizing models in a space parameterized by natural language is presented
- Using standard neural encoder–decoder components to build models for representation and search in this space, they demonstrated that new approach outperforms strong baselines on classification, structured prediction and reinforcement learning tasks.

Conclusions

- Language encourages compositional generalization
- Language simplifies structured exploration
- Language can help learning

References

- **Learning with Latent Language**, Jacob Andreas, Dan Klein, Sergey Levine [https://arxiv.org/abs/1711.00482]