Learning with Latent Language

Jacob Andreas Dan Klein Sergey Levine
Computer Science Division
University of California, Berkeley
{jda,klein,svlevine}@eecs.berkeley.edu

Abstract

The named concepts and compositional operators present in natural language provide a rich source of information about the abstractions humans use to navigate the world. Can this linguistic background knowledge improve the generality and efficiency of learned classifiers and control policies? This paper aims to show that using the space of natural language strings as a *parameter* space is an effective way to capture natural task structure. In a pretraining phase, we learn a language interpretation model that transforms inputs (e.g. images) into outputs (e.g. labels) given natural language descriptions. To learn a new concept (e.g. a classifier), we search directly in the space of descriptions to minimize the interpreter's loss on training examples. Crucially, our models do not require language data to learn these concepts: language is used only in pretraining to impose structure on subsequent learning. Results on image classification, text editing, and reinforcement learning show that, in all settings, models with a linguistic parameterization outperform those without.[1]

1 Introduction

The structure of natural language reflects the structure of the world. For example, the fact that it is easy for humans to communicate the concept *left of the circle* but comparatively difficult to communicate *mean saturation of the first five pixels in the third column* reveals something about the abstractions we find useful for interpreting and navigating our environment (Gopnik and Meltzoff, 1987). In machine learning, efficient automatic discovery of reusable abstract structure remains a major challenge. This paper investigates whether

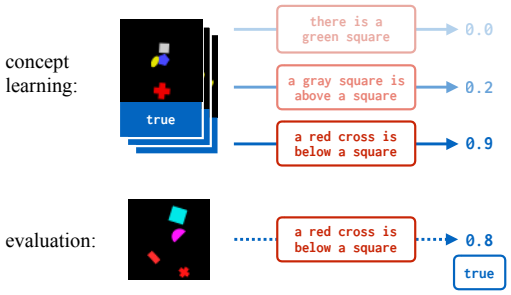


Figure 1: Example of our approach on a binary image classification task. We assume access to a pretrained language interpretation model that outputs the probability that an image matches a given description. To learn a new visual concept, we search in the space of natural language descriptions to maximize the interpretation model's score (top). The chosen description can be used with the interpretation model to classify new images (bottom). Figures best viewed in color.

background knowledge from language can provide a useful scaffold for acquiring it. We specifically propose to use language as a latent *parameter space* for few-shot learning problems of all kinds, including classification, transduction and policy search. We aim to show that this linguistic parameterization produces models that are both more accurate and more interpretable than direct approaches to few-shot learning.

Like many recent frameworks for multitask- and meta-learning, our approach consists of three phases: a pretraining phase, a concept-learning phase, and an evaluation phase. Here, the product of pretraining is a language interpretation model that maps from descriptions to predictors (e.g. image classifiers or reinforcement learners). Our thesis is that language learning is a powerful, general-purpose kind of pretraining, even for tasks that do not directly involve language.

New concepts are learned by searching directly in the space of natural language strings to mini-

[1]Code and data are available at https://github.com/jacobandreas/l3.

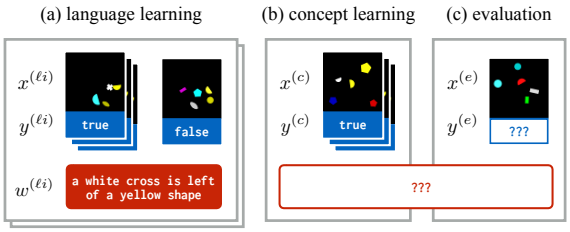


Figure 2: Formulation of the learning problem. Ultimately, we care about our model's ability to learn a concept from a small number of training examples (b) and successfully generalize it to held-out data (c). In this paper, concept learning is supported by a language learning phase (a) that makes use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

mize the loss incurred by the language interpretation model (Figure 1). Especially on tasks that require the learner to model high-level compositional structure shared by training examples, natural language hypotheses serve a threefold purpose: they make it easier to discover these compositional concepts, harder to overfit to few examples, and easier for humans to understand inferred patterns.

Our approach can be implemented using a standard kit of neural components, and is simple and general. In a variety of settings, we find that the structure imposed by a natural-language parameterization is helpful for efficient learning and exploration. The approach outperforms both multitask- and meta-learning approaches that map directly from training examples to outputs by way of a real-valued parameterization, as well as approaches that make use of natural language annotations as an additional supervisory signal rather than an explicit latent parameter. The natural language concept descriptions inferred by our approach often agree with human annotations when they are correct, and provide an interpretable debugging signal when incorrect. In short, by equipping models with the ability to "think out loud" when learning, they become both more comprehensible and more accurate.

2 Background

Suppose we wish to solve an image classification problem like the one shown in Figure 2b–c, mapping from images x to binary labels y. One straightforward way to do this is to solve a learning problem of the following form:

$$\arg\min_{\eta \in \mathbf{H}} \sum_{(x,\, y)} L(f(x;\, \eta),\, y)\,, \qquad (1)$$

where L is a loss function and f is a richly-parameterized class of models (e.g. convolutional networks) indexed by η (e.g. weight matrices) that map from images to labels. Given a new image x', $f(x';\, \eta)$ can be used to predict its label.

In the present work, we are particularly interested in *few-shot* learning problems where the number of $(x,\, y)$ pairs is small—on the order of five or ten examples. Under these conditions, directly solving Equation 1 is a risky proposition—any model class powerful enough to capture the true relation between inputs and outputs is also likely to overfit. For few-shot learning to be successful, extra structure must be supplied to the learner. Existing approaches obtain this structure by either carefully structuring the hypothesis space or providing the learner with alternative training data. The approach we present in this paper combines elements of both, so we begin with a review of existing work.

(Inductive) **program synthesis** approaches (e.g. Gulwani, 2011) reduce the effective size of the hypothesis class $\mathbf{H}$ by moving the optimization problem out of the continuous space of weight vectors and into a discrete space of formal program descriptors (e.g. regular expressions or Prolog queries). Domain-specific structure like version space algebras (Lau et al., 2003) or type systems (Kitzelmann and Schmid, 2006) can be brought to bear on the search problem, and the bias inherent in the syntax of the formal language provides a strong prior. But while program synthesis techniques are powerful, they are also limited in their application: a human designer must hand-engineer the computational primitives necessary to compactly describe every learnable hypothesis. While reasonable for some applications (like string editing), this is challenging or impossible for others (like computer vision).

An alternative class of **multitask learning** approaches (Caruana, 1998) import the relevant structure from other learning problems rather than defining it manually (Figure 2a, top). Since we may not know *a priori* what set of learning problems we ultimately wish to evaluate on, it is useful to think of learning as taking places in three phases:

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)

In these approaches, learning operates over two collections of parameters: shared parameters η and task-specific parameters θ. In pretraining, multitask approaches find:

$$\underset{\eta \in \mathbb{R}^a,\, \theta^{(\ell i)} \in \mathbb{R}^b}{\arg\min} \quad \sum_{i,j} L\big(f(x_j^{(\ell i)};\, \eta,\, \theta^{(\ell i)}),\, y_j^{(\ell i)}\big)\,. \quad (2)$$

At concept learning time, they solve for:

$$\underset{\theta^{(c)} \in \mathbb{R}^b}{\arg\min} \quad \sum_{j} L\big(f(x_j^{(c)};\, \eta,\, \theta^{(c)}),\, y_j^{(c)}\big) \quad (3)$$

on the new dataset, then make predictions for new inputs using $f(x^{(e)};\, \eta,\, \theta^{(c)})$.

Closely related **meta-learning** approaches (e.g. Schmidhuber, 1987; Santoro et al., 2016; Vinyals et al., 2016) make use of the same data, but collapse the inner optimization over $\theta^{(c)}$ and prediction of $y^{(e)}$ into a single learned model.

3 Learning with Language

In this work, we are interested in developing a learning method that enjoys the benefits of both approaches. In particular, we seek an intermediate language of task representations that, like in program synthesis, is both expressive and compact, but like in multitask approaches is learnable directly from training data without domain engineering. We propose to use natural language as this intermediate representation. We call our approach **learning with latent language** (L^3).

Natural language shares many structural advantages with the formal languages used in synthesis approaches: it is discrete, has a rich set of compositional operators, and comes equipped with a natural description length prior. But it also has a considerably more flexible semantics. And crucially, plentiful annotated data exists for *learning* this semantics: we cannot hand-write a computer program to recognize a *small dog*, but we can learn how to do it from image captions. More basically, the set of primitive operators available in language provides a strong prior about the kinds of abstractions that are useful for natural learning problems.

Concretely, we replace the pretraining phase above with a **language-learning** phase. We assume that at language-learning time we have access to natural-language **descriptions** $w^{(\ell i)}$ (Figure 2a, bottom). We use these w as *parameters*, in place of the task-specific parameters θ—that is, we learn a language **interpretation** model $f(x;\, \eta,\, w)$ that uses shared parameters η to turn a description w into a function from inputs to outputs. For the example in Figure 2, f might be an image rating model (Socher et al., 2014) that outputs a scalar judgment y of how well an image x matches a caption w.

Because these natural language parameters are observed at language-learning time, we need only learn the real-valued shared parameters η used for their interpretation (e.g. the weights of a neural network that implements the image rating model):

$$\underset{\eta \in \mathbb{R}^a}{\arg\min} \quad \sum_{i,j} L\big(f(x_j^{(\ell i)};\, \eta,\, w^{(\ell i)}),\, y_j^{(\ell i)}\big)\,. \quad (4)$$

At concept-learning time, conversely, we solve only the part of the optimization problem over natural language strings:

$$\underset{w^{(c)} \in \Sigma^*}{\arg\min} \quad \sum_{j} L\big(f(x_j^{(c)};\, \eta,\, w^{(c)}),\, y_j^{(c)}\big)\,. \quad (5)$$

This last step presents something of a challenge. When solving the corresponding optimization problem, synthesis techniques can exploit the algebraic structure of the formal language, while end-to-end learning approaches take advantage of differentiability. Here we can't do either—the language of strings is discrete, and any structure in the interpretation function is wrapped up inside the black box of f. Inspired by related techniques aimed at making synthesis more efficient (Devlin et al., 2017), we use learning to help us develop an effective optimization procedure for natural language parameters.

In particular, we simply 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:

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

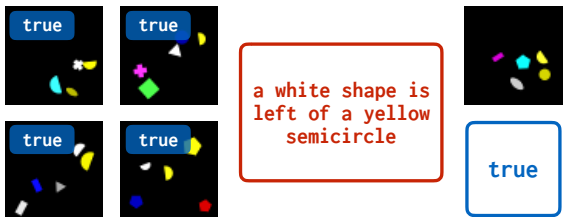


Figure 3: The few-shot image classification task. Learners are shown four positive examples of a visual concept (left) and must determine whether a fifth image matches the pattern (right). Natural language annotations are provided during language learning but must be inferred for concept learning.

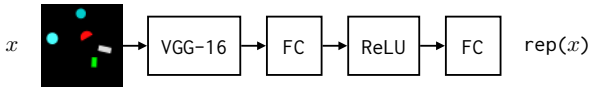
where q provides a (suitably normalized) approximation to the distribution of descriptions given task data. In the running example, this proposal distribution is essentially an image captioning model (Donahue et al., 2015). By sampling from q, we expect to obtain candidate descriptions that are likely to obtain small loss. But our ultimate inference criterion is still the true model f: at evaluation time we perform the minimization in Equation 5 by drawing a fixed number of samples, selecting the hypothesis $w^{(c)}$ that obtains the lowest loss, and using $f(x^{(e)}; \eta, w^{(c)})$ to make predictions.

What we have described so far is a generic procedure for equipping collections of related learning problems with a natural language hypothesis space. In Sections 4 and 5, we describe how this procedure can be turned into a concrete algorithm for supervised classification and sequence prediction. In Section 6, we describe how to extend these techniques to reinforcement learning.

4 Few-shot Classification

We begin by investigating whether natural language can be used to support high-dimensional few-shot classification. Our focus is on visual reasoning tasks like the one shown in Figure 3. In these problems, the learner is presented with four images, all positive examples of some visual concept like *a blue shape near a yellow triangle*, and must decide whether a fifth, held-out image matches the same concept. These kinds of reasoning problems have been well-studied in visual question answering settings (Johnson et al., 2017; Suhr et al., 2017). Our version of the problem, where the input and output feature no text data, but an explanation must be inferred, is similar to the visual reasoning problems proposed by Raven (1936) and Bongard (1968).

To apply the recipe in Section 2, we need to specify an implementation of the interpretation model f and the proposal model q. We begin by computing representations of input images x. We start with a pre-trained 16-layer VGGNet (Simonyan and Zisserman, 2014). Because spatial information is important for these tasks, we extract a feature representation from the final convolutional layer of the network. This initial featurization is passed through two fully-connected layers to form a final image representation, as follows:



We define interpretation and proposal models:[2]

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

The interpretation model f outputs the probability that x is assigned a positive class label, and is trained to maximize log-likelihood. Because only positive examples are provided in each language learning set, the proposal model q can be defined in terms of inputs alone. Details regarding training hyperparameters, RNN implementations, etc. may be found in Appendix A.

Our evaluation aims to answer two questions. First, does the addition of language to the learning process provide any benefit over ordinary multitask or meta-learning? Second, is it specifically better to use language as a hypothesis space for concept learning rather than just an additional signal for pretraining? We use several baselines to answer these questions:

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]

[2] Suppressing shared parameters η and λ for clarity.

[3] Many state-of-the-art approaches to meta-learning for classification (e.g. Snell et al., 2017) are not well-defined for possibly-overlapping evaluation classes with only positive examples provideded. Here we have attempted to provide a robust implementation that is as close as possible to the other systems under evaluation.

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

We report results on a dataset derived from the ShapeWorld corpus of Kuhnle and Copestake (2017). In this dataset the held-out image matches the target concept 50% of the time. In the validation and test folds, half of learning problems feature a concept that also appears in the language learning set (but with different exemplar images), while the other half feature both new images and a new concept. Images contain two or three distractor shapes unrelated to the objects that define the target concept. Captions in this dataset were generated from DMRS representations using an HPS grammar (Copestake et al., 2016). (Our remaining experiments use human annotators.) The dataset contains a total of 9000 pretraining tasks and 1000 of each validation and test tasks. More dataset statistics are provided in Appendix B.

Results are shown in Table 1. It can be seen that L^3 provides consistent improvements over the baselines, and that these improvements are present both when identifying new instances of previously-learned concepts and when discovering new ones. Some example model predictions are shown in Figure 4. The model often succeeds in making correct predictions, even though its inferred descriptions rarely match the ground truth. Sometimes this is because of inherent ambiguity in the description language (Figure 4a), and sometimes because the model is able to rule out candidates on the basis of partial captions alone (Figure 4b, where it is sufficient to recognize that the target concept involves a *circle*). More examples are provided in Appendix C.

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 respectively previously-used and novel visual concepts. 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 shows results when the model is given a ground-truth concept description rather than having to infer it from examples.

5 Programming by Demonstration

Next we explore whether the same technique can be applied to tasks that involve more than binary similarity judgments. We focus on structured prediction: specifically a family of string processing tasks. In these tasks, the model is presented with examples of five strings transformed according to some rule; it must then apply an appropriate transformation to a sixth (Figure 5). Learning proceeds as in the previous section, with:

$$\texttt{rep}(x, y) = \texttt{rnn-encode}([x, y])$$
$$f(y \mid x; w) =$$
$$\quad \texttt{rnn-decode}\big(y \mid [\texttt{rnn-encode}(x), \texttt{rnn-encode}(w)]\big)$$
$$q(w \mid \{(x_j, y_j)\}) =$$
$$\quad \texttt{rnn-decode}\big(w \mid \tfrac{1}{n}\textstyle\sum_j \texttt{rep}(x_j, y_j)\big)$$

Baselines are analogous to those for classification.

While string editing tasks of the kind shown in Figure 5 are popular in both the programming by demonstration literature (Singh and Gulwani, 2012) and the semantic parsing literature (Kushman and Barzilay, 2013), we are unaware of any datasets that support both learning paradigms at the same time. We have thus created a new dataset of string editing tasks by (1) sampling random regular transducers, (2) applying these transducers to collections of dictionary words, and (3) showing the collected examples to Mechanical Turk users

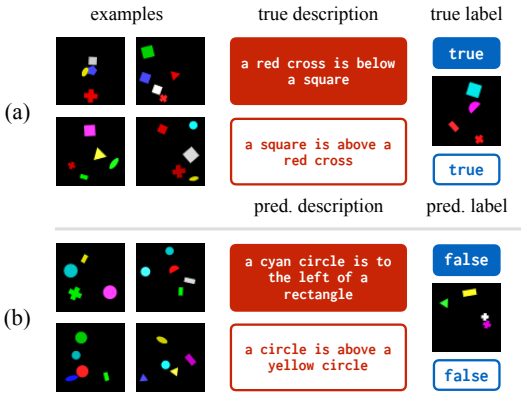


Figure 4: Example predictions for image classification. The model achieves high accuracy even though predicted descriptions rarely match the ground truth. High-level structure like the presence of certain shapes or spatial relations is consistently recovered.

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 model; using language data for joint training rather than as a hypothesis space provides little benefit.

and asking them to provide a natural language explanation with their best guess about the underlying rule. The dataset thus features both multi-example learning problems, as well as structured and unstructured annotations for each target concept. There are 3000 tasks for language learning and 500 tasks for each of validation and testing (Appendix B). Annotations are included in the code release for this paper.

Results are shown in Table 2. In these experiments, all models that use descriptions have been trained on the natural language supplied by human annotators. While we did find that the Meta+Joint model converges considerably faster than all the others, its final performance is somewhat lower than the baseline Meta model. As before, L^3 outperforms alternative approaches for learning directly from examples with or without descriptions.

Because all of the transduction rules in this dataset were generated from known formal descriptors, these tasks provide an opportunity to perform additional analysis comparing natural language to more structured forms of annotation (since we have access to ground-truth regular expressions) and more conventional synthesis-based methods (since we have access to a ground-truth regular expression execution engine). We additionally investigate the effect of the number of

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 or regular expressions.

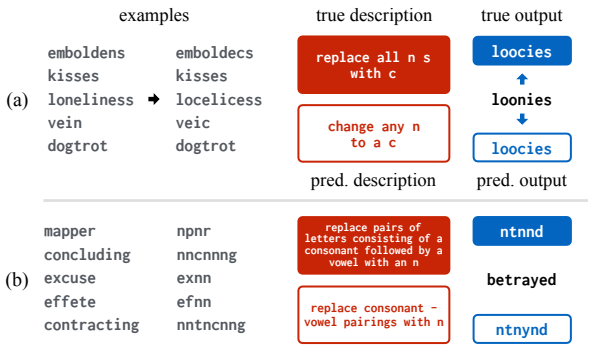


Figure 6: Example predictions for string editing.

samples drawn from the proposal model. These results are shown in Table 3.

A few interesting facts stand out. Under the ordinary evaluation condition (with no ground-truth annotations provided), language-learning with natural language data is actually better than language-learning with regular expressions. This might be because the extra diversity helps the model determine the relevant axes of variation and avoid overfitting to individual strings. Allowing the model to do its own inference is also better than providing ground-truth natural language descriptions, suggesting that it is actually better at generalizing from the relevant concepts than our human annotators (who occasionally write things like *I have no idea* for the inferred rule). Unsurprisingly, with ground truth REs (which unlike the human data are always correct) we can do better than any of the models that require inference. Coupling our inference procedure with an oracle RE evaluator, we essentially recover the synthesis-based approach of Devlin et al. (2017). Our findings are consistent with theirs: when an exact execution engine is available, there is no reason not to use it. But we can get almost 90% of the way there

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.

with an execution model learned from scratch. Examples of model behavior are shown in Figure 6; more may be found in Appendix D.

6 Policy Search

The previous two sections examined supervised settings where the learning signal comes from few examples but is readily accessible. In this section, we move to a set of reinforcement learning problems, where the learning signal is instead sparse and time-consuming to obtain. We evaluate on a collection of 2-D treasure hunting tasks. These tasks require the agent to discover a rule that determines the location of buried treasure in a large collection of environments of the kind shown in Figure 7. To recover the treasure, the agent must navigate (while avoiding water) to its goal location, then perform a `DIG` action. At this point the episode ends; if the treasure is located in the agent's current position, it receives a reward, otherwise it does not. In every task, the treasure has consistently been buried at a fixed position relative to some landmark (in Figure 7 a heart). Both the offset and the identity of the target landmark are unknown to the agent, and the location of the landmark varies across maps. Indeed, there is nothing about the agent's observations or action space to suggest that landmarks and offsets are even the relevant axes of variation across tasks: only the language reveals this structure.

The interaction between language and learning in these tasks is rather different from the supervised settings. In the supervised case, language serves mostly as a guard against overfitting, and

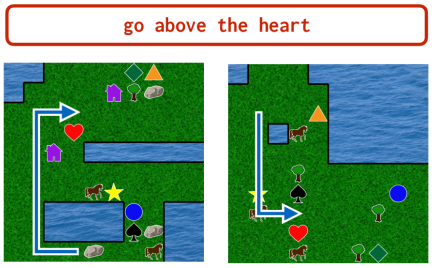
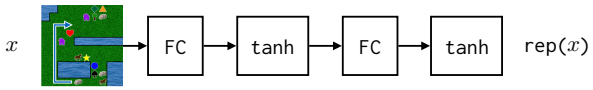


Figure 7: Example treasure hunting task: the agent is placed in a random environment and must collect a reward that has been hidden at a consistent offset with respect to some landmark. At language-learning time only, natural language instructions and expert policies are provided. The agent must both learn primitive navigation skills, like avoiding water, as well as the high-level structure of the reward functions for this domain.

can be generated conditioned on a set of pre-provided concept-learning observations. Here, agents are free to interact with the environment as much as they need, but receive observations only during interaction. Thus our goal here will be to build agents that can *adapt quickly* to new environments, rather than requiring them to immediately perform well on held-out data.

Why should we expect L^3 to help in this setting? In reinforcement learning, we typically encourage our models to explore by injecting randomness into either the agent's action space or its underlying parameterization. But most random policies exhibit nonsensical behaviors; as a result, it is inefficient both to sample in the space of network weights and to perform policy optimization from a random starting point. Our hope is that when parameters are chosen from within a structured family, a stochastic search in this structured space will only ever consider behaviors corresponding to a reasonable final policy, and in this way discover good behavior faster than ordinary RL.

Here the interpretation model f describes a policy that chooses actions conditioned on the current environment state and a linguistic parameterization. As the agent initially has no observations at all, we simply design the proposal model to generate unconditional samples from a prior over descriptions. Taking x to be an agent's current observation of the environment state, we define a state representation network and models:



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

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

This parameterization assumes a discrete action space, and assigns to each action a probability proportional to a bilinear function of the encoded description and world state. f is an instruction following model of a kind well-studied in natural language processing (Branavan et al., 2009); the proposal model allows it to generate its own instructions without external direction. To learn, we sample a fixed number of descriptions w from q. For each description, we sample multiple rollouts of the policy it induces to obtain an estimate of its average reward. Finally, we take the highest-scoring description and fine-tune its induced policy.

At language-learning time, we assume access to both natural language descriptions of these tar-

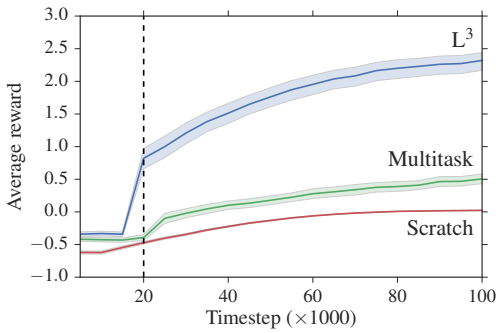


Figure 8: Treasure hunting reward obtained by each learning algorithm across multiple evaluation environments, after language learning has already taken place (bands show 95% confidence intervals for mean performance). *Multitask* learns an embedding for each task, while *Scratch* trains on every task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

get locations provided by human annotators, as well as expert policies for navigating to the location of the treasure. The multitask model we compare to replaces these descriptions with trainable task embeddings.[4] The learner is trained from task-specific expert policies using DAgger (Ross et al., 2011) during the language-learning phase, and adapts to individual environments using "vanilla" policy gradient (Williams, 1992) during the concept learning phase.

The environment implementation and linguistic annotations are in this case adapted from a natural language navigation dataset originally introduced by Janner et al. (2017). In our version of the problem (Figure 7), the agent begins each episode in a random position on a randomly-chosen map and must attempt to obtain the treasure. Relational concepts describing target locations are reused between language learning and concept-learning phases, but the environments themselves are distinct. For language learning the agent has access to 250 tasks, and is evaluated on an additional 50.

Averaged learning curves for held-out tasks are shown in Figure 8. As expected, reward for the L^3 model remains low during the initial exploration period, but once a description is chosen the score improves rapidly. Immediately L^3 achieves better reward than the multitask baseline, though it is not perfect; this suggests that the interpretation model is somewhat overfit to the pretraining environments. After fine-tuning even better results are rapidly obtained. Example rollouts are visualized in Appendix E. These results show that the model has used the structure provided by language to *learn* a better representation space for policies—one that facilitates sampling from a distribution over interesting and meaningful behaviors.

[4] In RL, the contribution of L^3 is orthogonal to that of meta-learning—one could use a technique like RL^2 (Duan et al., 2016) to generate candidate descriptions more efficiently, or MAML (Finn et al., 2017) rather than zero-shot reward as the training criterion for the interpretation model.

7 Other Related Work

This is the first approach we are aware of to frame a general learning problem as optimization over a space of natural language strings. However, many closely related ideas have been explored in the literature. String-valued latent variables are widely used in language processing tasks ranging from morphological analysis (Dreyer and Eisner, 2009) to sentence compression (Miao and Blunsom, 2016). Natural language annotations have been used in conjunction with training examples to guide the discovery of logical descriptions of concepts (Ling et al., 2017; Srivastava et al., 2017), and used as an auxiliary loss for training (Frome et al., 2013), analogously to the Meta+Joint baseline in this paper. Structured language-like annotations have been used to improve learning of generalizable structured policies (Oh et al., 2017; Andreas et al., 2017; Denil et al., 2017). Finally, natural language instructions available at *concept-learning* time (rather than language-learning time) have been used to provide side information to reinforcement learners about high-level strategy (Branavan et al., 2011), environments (Narasimhan et al., 2017) and exploration (Harrison et al., 2017).

8 Conclusion

We have presented an approach for learning in a space parameterized by natural language. Using simple models for representation and search in this space, we demonstrated that our approach outperforms standard baselines on classification, structured prediction and reinforcement learning tasks. We believe that these results suggest the following general conclusions:

Language encourages compositional generalization. Standard deep learning architectures are good at recognizing new instances of familiar

concepts, but not always at generalizing to new ones. By forcing decisions to pass through a linguistic bottleneck in which the underlying compositional structure of concepts is explicitly expressed, stronger generalization becomes possible.

Language simplifies structured exploration. Natural language scaffolding provides dramatic advantages in problems like reinforcement learning that require exploration: models with latent linguistic parameterizations can limit exploration to a class of behaviors that are likely *a priori* to be goal-directed and interpretable.

And generally, *language can help learning*. In multitask settings, it can even improve learning on tasks for which no language data is available at training or test time. While some of these advantages are also provided by techniques built on top of formal languages, natural language is at once more expressive and easier to obtain than formal supervision. We believe this work hints at broader opportunities for using naturally-occurring language data to improve machine learning for tasks of all kinds.

Acknowledgments

JA is supported by a Facebook graduate fellowship.

References

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the International Conference on Machine Learning*.

Mikhail Moiseevich Bongard. 1968. The recognition problem. Technical report.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

S.R.K. Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a Monte-Carlo framework. In *Proceedings of the Human Language Technology Conference of the Association for Computational Linguistics*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Ann A Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszynska. 2016. Resources for building applications with dependency minimal recursion semantics. In *Language Resources and Computation*.

Misha Denil, Sergio Gómez Colmenarejo, Serkan Cabi, David Saxton, and Nando de Freitas. 2017. Programmable agents. *arXiv preprint arXiv:1706.06383* .

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. 2017. RobustFill: Neural program learning under noisy I/O. In *Proceedings of the International Conference on Machine Learning*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* .

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*.

Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, MarcAurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*.

Alison Gopnik and Andrew Meltzoff. 1987. The development of categorization in the second year and its relation to other cognitive and linguistic developments. *Child Development* .

Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM SIGPLAN Notices* 46(1).

Brent Harrison, Upol Ehsan, and Mark O Riedl. 2017. Guiding reinforcement learning exploration using natural language. *arXiv preprint arXiv:1707.08616* .

Michael Janner, Karthik Narasimhan, and Regina Barzilay. 2017. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics* .

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622* .

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *Proceedings of the Conference on Computer Vision and Pattern Recognition* .

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Emanuel Kitzelmann and Ute Schmid. 2006. Inductive synthesis of functional programs: An explanation based generalization approach. *Journal of Machine Learning Research* 7.

Alexander Kuhnle and Ann Copestake. 2017. Shapeworld-a new test methodology for multimodal language understanding. *arXiv preprint arXiv:1704.04517* .

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Tessa Lau, Steven A Wolfman, Pedro Domingos, and Daniel S Weld. 2003. Programming by demonstration using version space algebra. *Machine Learning* 53(1).

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2017. Deep transfer in reinforcement learning by language grounding. *arXiv preprint arXiv:1708.00133* .

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

John C Raven. 1936. Mental tests used in genetic studies: The performance of related individuals on tests mainly educative and mainly reproductive. *Unpublished masters thesis, University of London* .

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of the International Conference on Machine Learning*.

Jurgen Schmidhuber. 1987. Evolutionary principles in self-referential learning. *On learning how to learn. Diploma thesis, Institut f. Informatik, Tech. Univ. Munich* .

K Simonyan and A Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arxiv:1409.1556* .

Rishabh Singh and Sumit Gulwani. 2012. Learning semantic string transformations from examples. In *International Conference on Very Large Databases*.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175* .

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics* 2.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *55th Annual Meeting of the Association for Computational Linguistics, ACL*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4).

A Model and Training Details

In all models, RNN encoders and decoders use gated recurrent units (Cho et al., 2014).

Few-shot classification Models are trained with the ADAM optimizer (Kingma and Ba, 2015) with a step size of 0.0001 and batch size of 100. The number of pretraining iterations is tuned based on subsequent concept-learning performance on the development set. Neural network hidden states, task parameters, and word vectors are all of size 512. 10 hypotheses are sampled during for each evaluation task in the concept-learning phase.

Programming by demonstration Training as in the classification task, but with a step size of 0.001. Hidden states are of size 512, task parameters of size 128 and word vectors of size 32. 100 hypotheses are sampled for concept learning.

Policy search DAgger (Ross et al., 2011) is used for pre-training and vanilla policy gradient (Williams, 1992) for concept learning. Both learning algorithms use ADAM with a step size of 0.001 and a batch size of 5000 samples. For imitation learning, rollouts are obtained from the expert policy on a schedule with probability 0.95^t (for t the current epoch). For reinforcement learning, a discount of 0.9 is used. Because this dataset contains no development data, pretraining is run until performance on the pretraining tasks reaches a plateau. Hidden states and task embeddings are of size 64. 100 hypotheses are sampled for concept learning, and 1000 episodes (divided evenly among samples) are used to estimate hypothesis quality before fine-tuning.

B Dataset Information

ShapeWorld This is the only fully-synthetic dataset used in our experiments. Each scene features 4 or 5 non-overlapping entities. Descriptions refer to spatial relationships between pairs of entities identified by shape, color, or both. There are 8 colors and 8 shapes. The total vocabulary size is only 30 words, but the dataset contains 2643 distinct captions. Descriptions are on average 12.0 words long.

Regular expressions Annotations were collected from Mechanical Turk users. Each user was presented with the same task as the learner in this paper: they observed five strings being transformed, and had to predict how to transform a sixth. Only after they correctly generated the held-out word were they asked for a description of the rule. Workers were additionally presented with hints like "look at the beginning of the word" or "look at the vowels". Descriptions are automatically preprocessed to strip punctuation and ensure that every character literal appears as a single token.

The regular expression data has a vocabulary of 1015 rules and a total of 1986 distinct descriptions. Descriptions are on average 12.3 words in length but as long as 46 words in some cases.

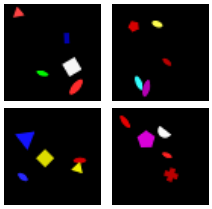
Navigation The data used was obtained from Janner et al. (2017). We created our own variant of the dataset containing collections of related tasks. Beginning with the "local" tasks in the dataset, we generated alternative goal positions at fixed offsets from landmarks as described in the main section of this paper. Natural-language descriptions were selected for each task collection from the human annotations provided with the dataset. The vocabulary size is 74 and the number of distinct hints 446. The original action space for the environment is also modified slightly: rather than simply reaching the goal cell (achieved with reasonably high frequency by a policy that takes random moves), we require the agent to commit to an individual goal cell and end the episode with a special `DIG` action.

Data augmentation Due to their comparatively small size, a data augmentation scheme (Jia and Liang, 2016) is employed for the regular expression and navigation datasets. In particular, wherever a description contains a recognizable entity name (i.e. a character literal or a landmark name), a description template is extracted. These templates are then randomly swapped in at training time on other examples with the same high-level semantics. For example, the description *replace first b with e* is abstracted to *replace first CHAR1 with CHAR2*, and can subsequently be specialized to, e.g., *replace first c with d*. This templating is easy to implement because we have access to ground-truth structured concept representations at training time. If these were not available it would be straightforward to employ an automatic template induction system (Kwiatkowski et al., 2011) instead.

C Examples: ShapeWorld

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

Positive examples:



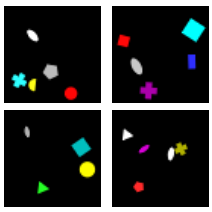
True description:
a red ellipse is to the right of an ellipse

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

Input:

True label: true

Pred. label: true

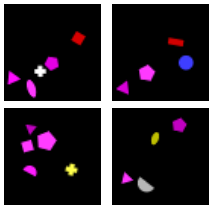


a shape is below a white ellipse

a white shape is to the left of a yellow ellipse

false

true

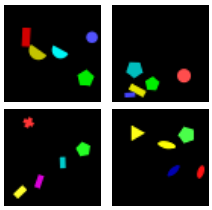


a magenta triangle is to the left of a magenta pentagon

a magenta triangle is to the left of a pentagon

true

true

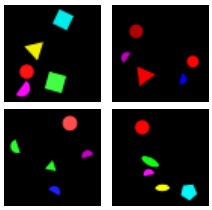


a green pentagon is to the right of a yellow shape

a green shape is to the right of a red semicircle

false

false

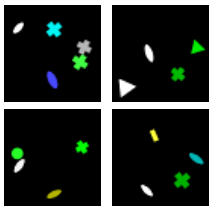


a red circle is above a magenta semicircle

a green triangle is above a red circle

false

true



a white ellipse is to the left of a green cross

a green cross is to the right of a white ellipse

true

true

D Examples: Regular Expressions

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

uptakes	uptakes	replace every o with u n		regulatiunn
pouching	punuching		regulation	
embroidery	embrunidery			
rebelliousness	rebelliunusness	change all o to u n		regulatinun
stoplight	stunplight			

fluffiest	fluffiest	the leter m is replaced by ee		chartering
kidnappers	kidnappers		chartering	
matting	eeatting			
griping	griping	change every m to ee		chartering
disagreements	disagreeeeents			

clandestine	clandqtine	e		gratuity
limning	limning			
homes	homq		gratuity	
lifeblood	lifqlood	where e appears , replace it		
inflates	inflatq	and the following letter with q		gratuity

fruitlessly	fruitlessly	if the word ends with an s , replace		promptemu
sandier	sandier	the last two letters with m u		
washers	washemu		prompters	
revelries	revelrimu			
dewlaps	dewlamu	change last to m u if consonant		promptemu

ladylike	ladylike	ending consonant is replaced with n l		initialnl
flintlocks	flintlocknl		initials	
student	studennl			
surtaxes	surtaxenl	drop last two and add n l		initialnl
bedecks	bedecknl			

porringer	porringeer	add e next to letter g		rage
puddling	puddlinge			
synagog	synageoge		rag	
curtseying	curtseyinge	when a letter is preceded by a g ,		
monsieur	monsieur	e is added after that letter		rage

trivializes	trivializes	replace the 1st 2 letters of the word with a g x		
tried	tried	if the word begins with a consonant then a vowel		gxndlords
tearfully	gxarfully		landlords	
hospitalize	gxspitalize			
patronizing	gxtronizing	if the second letter is a vowel , replace the		gxndlords
		first two letters with g x		

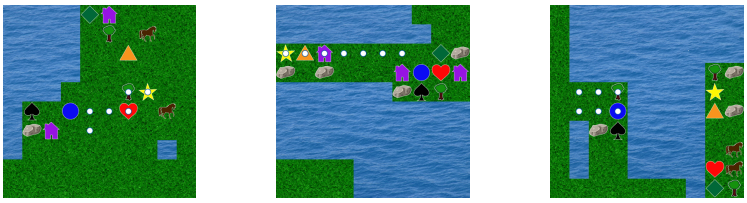
microseconds	microsecnyr	replace consonants with y r		exertion
antiviral	antiviral			
flintlock	flintloyr		exertion	
appreciable	appreciabyr			
stricter	stricter	the last two letters are replaced by y r		exertiyr

E Examples: Navigation

White breadcrumbs show the path taken by the agent.

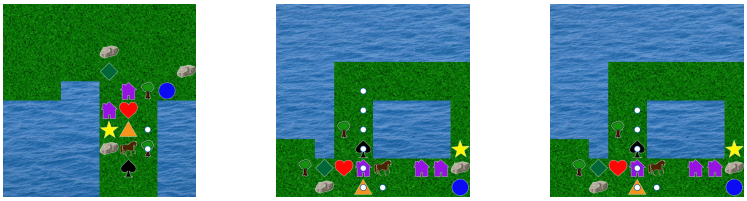
Human description:
`move to the star`

Inferred description:
`reach the star cell`



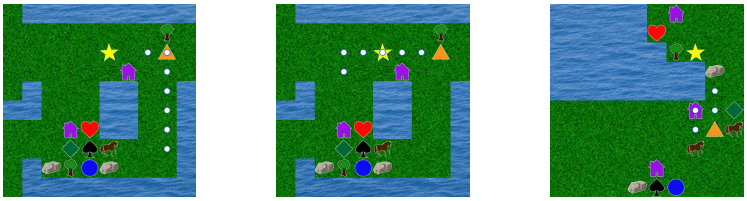
`reach square one right of triangle`

`reach cell to the right of the triangle`



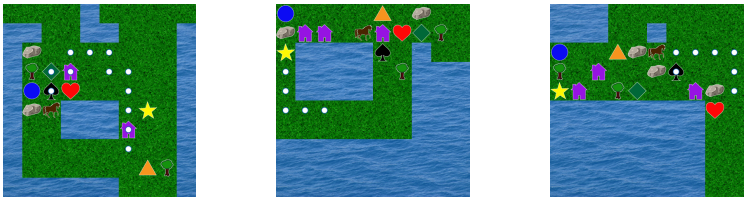
`reach cell on left of triangle`

`reach square left of triangle`



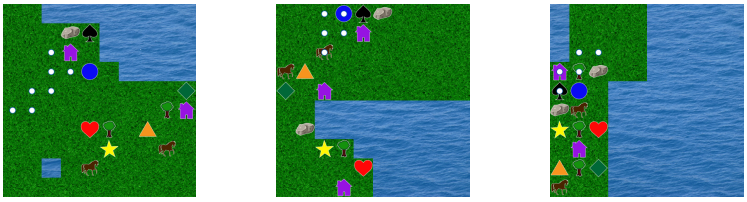
`reach spade`

`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`