A Proposal for Learning Compositional Problem-Solvers

Michael B. Chang[*], **Abhishek Gupta**[*], **Sergey Levine**[*], and **Thomas L. Griffiths**[**]
[*]U.C. Berkeley
[**]Princeton University
[*]{mbchang,abhigupta,svlevine}@eecs.berkeley.edu
[**]tomg@princeton.edu

Abstract

Assuming compositionality may enable a learner to exploit structure in a problem space to extrapolate to "out-of-distribution" problems by casting the extrapolation problem as one of learning algorithmic procedures over transformations between representations. This paper introduces the *compositional learner*, a possible way to learn *what* primitive transformations to compose and *how* to compose them. It views such a composition as a traversal through a meta-Markov decision process. Experiments on an illustrative task of multilingual arithmetic demonstrate that the compositional learner can discover a hierarchical decompositional of a problem into its subproblems and extrapolate beyond the training distribution to a more complex disjoint distribution of problems, yielding a 10-fold reduction in sample complexity compared to a monolithic recurrent neural network. It can also compose learned spatial transformations to recover canonical MNIST digits.

1 Introduction

Continual learning is largely a question of organizing previously-learned knowledge in a way that makes solving future unknown problems *easy*. Because problem-solving involves extrapolating to problems outside of the empirical problem distribution, these problems appear out-of-distribution to *flat learners* that follow the traditional neural network paradigm of learning input-output mappings. In general, meta-learning flat learners [4, 15, 16, 19–21, 28–30, 33, 43] sidestep this challenge by allowing the learner to train on the new problem distribution, but the original challenge of extrapolation still remains. This paper proposes that an out-of-distribution problem from a flat learner's perspective can be transformed into an in-distribution one from a compositional learner's perspective by exploiting the compositional structure of the problem distribution, such that extrapolation comes for free without additional training.

Sec. 2 represents a compositionally-structured multi-task distribution as a *problem graph*, which motivates compositional learners as a natural alternative to flat learners. Sec. 3 introduces a *compositional learner* as a collection of specialized *computational modules*: atomic function operators that perform transformations between representations. These modules, each one of which is learned from the empirical distribution, can then be composed to iteratively re-represent an out-of-distribution problem into a in-distribution one, thereby providing a possible solution to the extrapolation problem.

2 The compositional nature of problem-solving

Humans navigate foreign cities and understand novel conversations despite only observing a tiny fraction of the true distribution of the world. A hypothesis for how it is even possible to extrapolate like so is that the world contains *compositional structure*, such that solving a novel problem is possible by composing previously learned partial solutions in a novel way to fit the context.

Continual Learning Workshop, 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

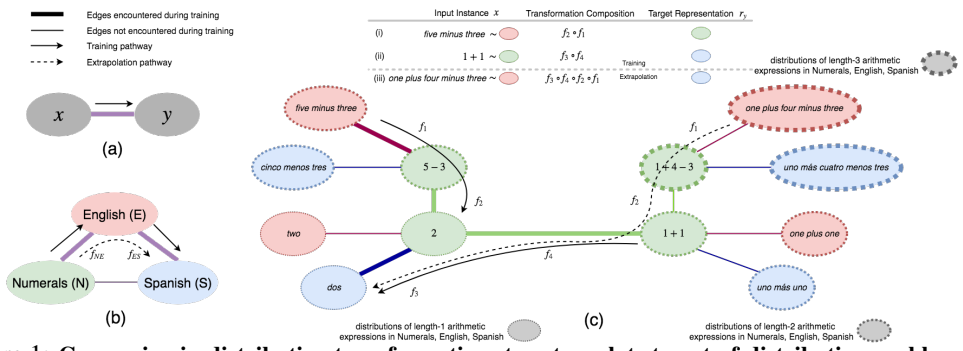


Figure 1: **Composing in-distribution transformations to extrapolate to out-of-distribution problems: (a) Standard supervised learning**: A *problem* requires the learner to map an instance x from one distribution r_x to its corresponding instance y in another distribution r_y. **(b) Multi-task learning**: During training the learner encounters 2 tasks $N \rightarrow E$ and $E \rightarrow S$ so both transformations are *in-distribution*. A *flat learner* learns a single network for both tasks, so $N \rightarrow S$ is an *out-of-distribution* task. A *compositional learner* explicitly tries to distill out primitive transformations as computational modules, so it may compose previously learned modules f_{NE} and f_{ES} to extrapolate. **(c) An example problem graph for multilingual arithmetic**: The compositional learner has learned modules to translate English to numerals (f_1), subtract (f_2), translate numerals to Spanish (f_3), and add (f_4) on distributions of 1-length and 2-length multilingual arithmetic expressions. By composing together these functions, it extrapolates to solve a strictly harder a 3-length multilingual arithmetic expression involving a language pair it has not seen. These modules need not necessarily be directly supervised; the compositional learner may design modules for its own use that need not mirror the generative transformations that created the data. To avoid clutter not all possible generative transformation edges of the problem graph are shown, although none are more privileged than any other.

Assuming the real world is compositional suggests that the generative distribution of observations from the environment can be modeled as an irreducible *base distribution* of entities and a set of (not necessarily 1-to-1) *generative transformations* that compose together to transform the base distribution into different instantiations. Together, the base distribution and its various instantiations are the nodes of a (not necessarily fully) connected *problem graph* (Fig. 1), the edges connecting which are the generative transformations that convert between one instantiation to another. Let the distribution at a node be called a *representation* and a sample from the representation be called an *instance*. Then the instance from the base representation and its corresponding instances in other representations form an equivalence class, so the graph is a distribution over these equivalence classes.[1] A *problem* can be defined as a tuple (x, r_y), where x is an instance from a source representation r_x and r_y is a target representation. To solve a problem[2] is to transform x into y, where y is an instance equivalent to x from r_y. A *task* is a distribution of problems defined on r_x and r_y. For example, let the base distribution be over the parameters of a face, the set of generative transformations be "rotation" and "lighting", and the nodes of the problem graph be faces that have been rotated right, rotated right and lit from the left, lit from below, etc. A face and different transformations of which comprise an equivalence class. An example problem is to generate a given face with novel rotation and lighting conditions or decide whether two faces under different transformations are the same [27, 45].

A *flat learner*, trained to directly map $x \in r_x$ to $y \in r_y$, does not explicitly model the composition of generative transformations that form the path from r_x to r_y. A task (r_x, r_y) is *out-of-distribution* if that pair of representations was not observed with supervisory signal during training. For example, if the learner has received supervision for translating English to French (EF), and French to Spanish (FS), but not English to Spanish (ES), then ES is an out-of-distribution task and "EF" and "FS" are *in-distribution* tasks. Clearly, expecting a flat learner to extrapolate to out-of-distribution problems becomes unrealistic as the composition depth of generative transformations grows.

3 Compositional Learners

The *compositional learner* (Fig. 2) explicitly attempts to self-organize around the edges of the problem graph to make extrapolating to out-of-distribution problems possible by composing together

[1] Because of equivalence it does not matter which node represents the base distribution.

[2] "Solving a problem simply means representing it so as to make the solution transparent" Simon [42].

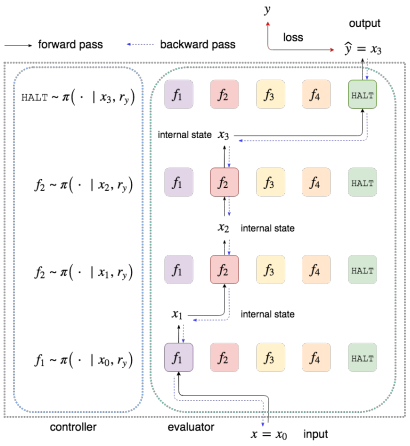


Figure 2: **Compositional learner:** The implementation consists of a controller $\pi(f|x, r_y)$, a set of modules f_k, and an evaluator. At step j, the controller observes the internal state x_j and the target output representation r_y. It selects a module f_k and a portion of x_j to operate on. The evaluator applies f_k to transform x_j into a new internal state x_{j+1}. When π selects `HALT`, a loss is computed by comparing the current internal state with the desired output. The loss is backpropagated through the modules, which are trained with Adam [26]. The controller receives a sparse reward derived from the loss, incurs a cost for each computation, and is trained with proximal policy optimization [40].

the modules that it learned to model in-distribution transformations during training. The application of a sequence of modules can be likened to the execution trace of a program, where a computation is the application of a module to an internal state. Learning such compositions can be formulated as a sequential decision-making problem in a meta-level Markov decision process (MDP) [22], where the state space is the learner's internal states of computation and the action space is the set of modules.

This meta-MDP is the the learner's own *internal* representation of the problem graph that it builds from the many tasks it encounters. Ideally, eventually its internal states and modules will converge to respectively mirror the representations and generative transformations in the *external* problem graph, which in general is unknown, by pattern-matching across tasks to extract shared *subproblems* whose solutions it can encapsulate in its modules.

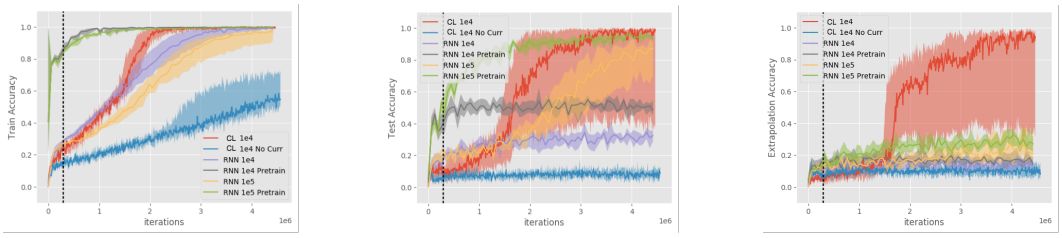
To encourage such compositionality and specialization to emerge requires constraints on the internal states and modules[3] because to transform an out-of-distribution problem to an in-distribution one requires some assumptions about the type of compositional structure in the task distribution. This paper contains experiments where the internal states live in the same vocabulary as the external representations, the modules are parametrized the same way as the generative transformations, and the learner is trained with a curriculum of tasks, equivalent to slowly growing the observable frontier of the problem graph, to encourage the learner to reuse previously learned modules for more complex tasks. Future work will investigate to what extent softer information-theoretic constraints can still encourage compositionality and specialization.

Given these constraints, the modules can learn to make local progress on solving their respective in-distribution subproblems, even though the global problem may be out-of-distribution. Due to the learner's random initialization, certain modules that initially perform better on and are initially chosen by the controller for certain subproblems are more likely to be later chosen (and hence trained) on these subproblems, which gradually pushes the modules towards specialization and the controller to reflect the compositional structure of the task distribution.

4 Experiments

Mulitlingual Arithmetic (Fig. 3): The learner trains on a curriculum of 46200 examples of 2, 3, 4, 5-length expressions ($2.76 \cdot 10^{-4}$ of the training distribution), and is asked to extrapolate to 5-length expressions and 10-length expressions with unseen language pairs (problem space: $4.92 \cdot 10^{15}$ problems). It achieves about 60% accuracy for extrapolating to 100-term problems (problem space: $4.29 \cdot 10^{148}$). Modules learn to perform arithmetic operations and return the result in a specific language, resulting in hybrid-language arithmetic expressions that highlight the difference between the learner's internal problem graph and the external one it is trained on. For extrapolation, the learner learns to first solve the expression into some language, then takes an additional step to translate the final answer into the desired language.

[3] Otherwise there is no reason for the compositional learner to degenerate into a flat learner where one modules attempts to learn everything. Technically, statistical learning theory does not guarantee out-of-distribution extrapolation at all; assuming compositionality warps the task distribution such that out-of-distribution problems for a flat-learner are in-distribution problems in disguise for a compositional learner.



(a) Training Accuracy (b) Length-5 Extrap. Accuracy (c) Length-10 Extrap. Accuracy

Figure 3: **Multilingual arithmetic.** The compositional learner (CL) extrapolates out of distribution significantly better than the RNN, which requires ten times more data to extrapolate to 5-length expressions and does not 10-length multilingual arithmetic expressions even with more data. Pretraining the RNN directly on the generative transformations does not help the 10-length case, highlighting a limitation of using flat learners for compositional problems. By comparing CL with a version trained without a curriculum ("No Curr": blue), we see the benefit of slowly growing the complexity of problems throughout training, although this benefit does not transfer to the RNN. The vertical black dashed line indicates at which point all the training data has been added when CL is trained with a curriculum (red). Note that generalization becomes apparent only after a million iterations after all the training data has been added. **(b, c)** only show accuracy on the expressions with the maximum length of those added so far to the curriculum. "1e4" and "1e5" correspond to the order of magnitude of the number of samples in the dataset, of which 70% are used for training. 10, 50, and 90 percentiles are shown over 6 runs.

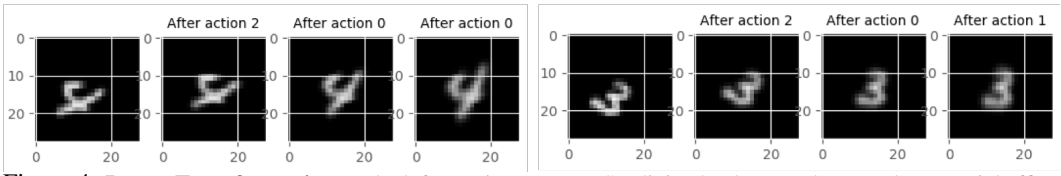


Figure 4: **Image Transformations:** The leftmost images MNIST digits that have undergone three spatial affine transformations. CL is constrained to take exactly three steps. This domain is challenging because the underlying compositional structure is latent; because of the weaker constraints on its internal states, CL sometimes falls into a trap of only choosing one module and get stuck in suboptimal local optima. **left:** CL translates, then rotates twice. **right:** CL translates, then rotates, then enlarges.

Image Transformations (Fig. 4): Suppose the learner has knowledge of what untransformed MNIST digits look like; is it possible to learn to compose appropriate spatial affine transformations in sequence to convert the transformed MNIST digit into a "canonical" one, such that it can use a pre-trained classifier to classify it? The learner is trained on a distribution of MNIST digits modified by either one or two spatial transformations (rotate, scale, translate) and is asked to transfer to images generated by composing three transformations. It is initalized with four types of modules: a Spatial Transformer Network (STN) [25] parametrized to only rotate (action 0), an STN that only scales (action 1), an STN that only translates (action 2), and an identity function (action 3). In contrast to multilingual arithmetic, constraints on the internal states are much weaker, making this domain much more difficult to optimize and a target for future work.

5 Discussion and Related Work

The problem graph is loosely inspired by Plato's theory of Forms [32], in which observable entities in the world are instantiations of the *essence* (Form) that underlies them. Viewing the problem graph as a generative probabilistic program establishes a relation with type systems, where representations are analogous to types. The modules that a compositional learner learns are analogous to the recognition weights of the Helmholtz Machine [9] and Neural Heat Exchanger [39]. Thus this learner may lead to a possible way to implement compositional amortized inference [36] over an unknown generative probabilistic program. This paper is an initial step towards bridging re-representation [2, 24, 41] with deep learning. Learning both the controller and modules relate this paper to hierarchical reinforcment learning [5]. This paper is a step toward combining the strengths of approaches to neural program induction that pre-specify modules and learn their composition [6–8, 11, 14, 17, 34, 44, 46] and vice versa [3, 10, 35]. In contrast with other self-organizing learners that learn to select experts [12, 31] or route through fixed architectures [13, 23, 37], this paper focuses on learning to compose computations similar to [1, 18], working toward a metareasoning approach [22, 38] for continual multitask learning.

References

[1] F. Alet, T. Lozano-Pérez, and L. P. Kaelbling. Modular meta-learning. *arXiv preprint arXiv:1806.10166*, 2018.

[2] J. R. Anderson. The adaptive character of thought. chapter 5, pages 191–230. Psychology Press, 1990.

[3] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

[4] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[5] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[6] R. Bunel, M. Hausknecht, J. Devlin, R. Singh, and P. Kohli. Leveraging grammar and reinforcement learning for neural program synthesis. *arXiv preprint arXiv:1805.04276*, 2018.

[7] J. Cai, R. Shin, and D. Song. Making neural programming architectures generalize via recursion. *arXiv preprint arXiv:1704.06611*, 2017.

[8] X. Chen, C. Liu, and D. Song. Learning neural programs to parse programs. *arXiv preprint arXiv:1706.01284*, 2017.

[9] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[10] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2169–2176. IEEE, 2017.

[11] S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2):7–52, 2001.

[12] K. T. Feigelis, B. Sheffer, and D. L. Yamins. Modular continual learning in a unified visual environment. *arXiv preprint arXiv:1711.07425*, 2017.

[13] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[14] J. K. Feser, M. Brockschmidt, A. L. Gaunt, and D. Tarlow. Differentiable functional program interpreters. *arXiv preprint arXiv:1611.01988*, 2016.

[15] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[16] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

[17] Y. Ganin, T. Kulkarni, I. Babuschkin, S. A. Eslami, and O. Vinyals. Synthesizing programs for images using reinforced adversarial learning. *arXiv preprint arXiv:1804.01118*, 2018.

[18] A. L. Gaunt, M. Brockschmidt, N. Kushman, and D. Tarlow. Differentiable programs with neural libraries. *arXiv preprint arXiv:1611.02109*, 2016.

[19] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[20] A. Gupta, B. Eysenbach, C. Finn, and S. Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

[21] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. Meta-reinforcement learning of structured exploration strategies. *arXiv preprint arXiv:1802.07245*, 2018.

[22] N. Hay, S. Russell, D. Tolpin, and S. E. Shimony. Selecting computations: Theory and applications. *arXiv preprint arXiv:1408.2048*, 2014.

[23] G. Hinton, N. Frosst, and S. Sabour. Matrix capsules with em routing. 2018.

[24] R. C. Holte and B. Y. Choueiry. Abstraction and reformulation in artificial intelligence. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358(1435):1197–1204, 2003.

[25] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4390–4399, 2015.

[28] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[29] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. 2018.

[30] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR, abs/1803.02999*, 2018.

[31] G. Parascandolo, M. Rojas-Carulla, N. Kilbertus, and B. Schölkopf. Learning independent causal mechanisms. *arXiv preprint arXiv:1712.00961*, 2017.

[32] Plato. *Republic*. 380 B.C.

[33] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. 2016.

[34] S. Reed and N. De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

[35] S. Riedel, M. Bosnjak, and T. Rocktäschel. Programming with a differentiable forth interpreter. *CoRR, abs/1605.06640*, 2016.

[36] D. Ritchie, P. Horsfall, and N. D. Goodman. Deep amortized inference for probabilistic programs. *arXiv preprint arXiv:1610.05735*, 2016.

[37] C. Rosenbaum, T. Klinger, and M. Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *International Conference on Learning Representations*, 2018.

[38] S. Russell and E. Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1-3):361–395, 1991.

[39] J. Schmidhuber. The neural heat exchanger. 1996.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[41] H. A. Simon. The sciences of the artificial. *Cambridge, MA*, page 132, 1969.

[42] H. A. Simon. The science of design: Creating the artificial. *Design Issues*, pages 67–82, 1988.

[43] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.

[44] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese. Neural task programming: Learning to generalize across hierarchical tasks. *arXiv preprint arXiv:1710.01813*, 2017.

[45] I. Yildirim, W. Freiwald, and J. Tenenbaum. Efficient inverse graphics in biological face processing. *bioRxiv*, page 282798, 2018.

[46] W. Zaremba, T. Mikolov, A. Joulin, and R. Fergus. Learning simple algorithms from examples. In *International Conference on Machine Learning*, pages 421–429, 2016.