Generating Diverse Programs with Instruction Conditioned Reinforced Adversarial Learning

Aishwarya Agrawal
Georgia Tech*
aishwarya@gatech.edu

Mateusz Malinowski
DeepMind
mateuszm@google.com

Felix Hill
DeepMind
felixhill@google.com

Ali Eslami
DeepMind
aeslami@google.com

Oriol Vinyals
DeepMind
vinyals@google.com

Tejas Kulkarni
DeepMind
tkulkarni@google.com

Abstract

Advances in Deep Reinforcement Learning have led to agents that perform well across a variety of sensory-motor domains. In this work, we study the setting in which an agent must learn to generate programs for diverse scenes conditioned on a given symbolic instruction. Final goals are specified to our agent via images of the scenes. A symbolic instruction consistent with the goal images is used as the conditioning input for our policies. Since a single instruction corresponds to a diverse set of different but still consistent end-goal images, the agent needs to learn to generate a distribution over programs given an instruction. We demonstrate that with simple changes to the reinforced adversarial learning [3] objective, we can learn instruction conditioned policies to achieve the corresponding diverse set of goals. Most importantly, our agent's stochastic policy is shown to more accurately capture the diversity in the goal distribution than a fixed pixel-based reward function baseline. We demonstrate the efficacy of our approach on two domains: (1) drawing MNIST digits with a paint software conditioned on instructions and (2) constructing scenes in a 3D editor that satisfies a certain instruction.

1 Introduction

Virtual worlds are being increasingly used to make advancements in deep reinforcement learning research [3, 1, 2, 7]. However, these worlds are typically hand-designed, and therefore their construction at scale is expensive. In this work, we build generative agents that output programs for diverse scenes that can be used to create virtual worlds, conditioned on an instruction. Thus, with a short text string, we can obtain a distribution over multiple diverse virtual scenes, each consistent with the instruction. We experiment with two instruction-conditioned domains: (1) drawing MNIST digits with a paint software, and (2) constructing scenes in a 3D editor; both are shown in Figure 1.

We build our agents on top of recently introduced reinforced adversarial learning framework [3], where final goals are specified to the agent via images of the virtual scenes. Unlike [3] where, either random noise (unconditional generation) or the goal image (reconstruction) is used as the conditioning input, in our work, a symbolic instruction is used as the conditioning input for our policies. Since, a single instruction corresponds to a diverse set of goal images, the agent needs to learn to generate a distribution over programs such that all programs are consistent with the instruction. A discriminator that takes an image (generated or an example goal image) and the corresponding instruction acts as a reward learning function to provide training signal to the generator. Thus, the

*This work was done while Aishwarya was interning at DeepMind.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

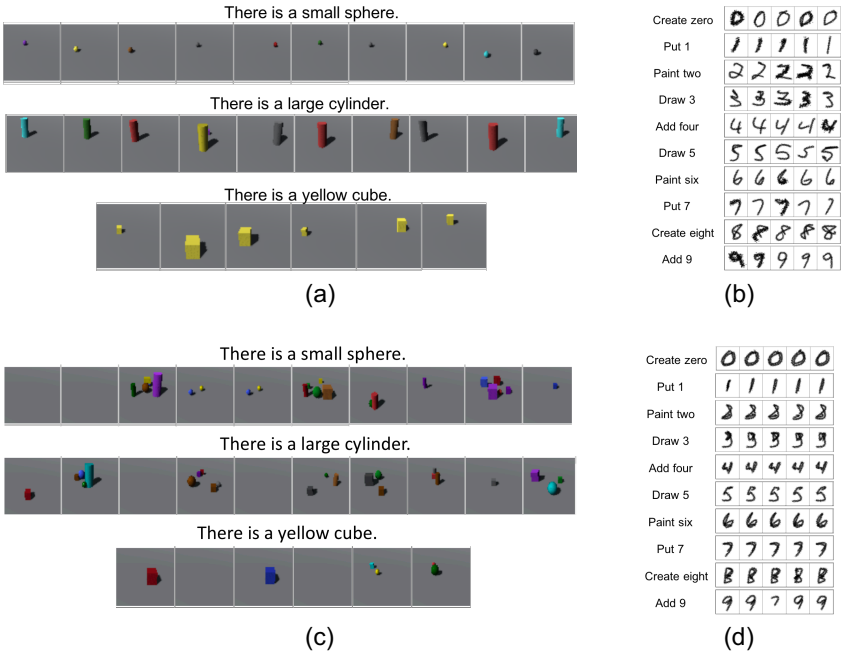

Figure 1: We build agents that can generate programs for diverse scenes conditioned on a given symbolic instruction. (a) Conditioned on a given instruction (e.g., "There is a small sphere".), the agent learns to generate programs for 3D scenes such that all scenes have a small sphere. We can see that the generated scenes show diversity in unspecified attributes (color and location of the sphere). (b) Conditioned on a simple instruction (e.g., "Draw zero", "Paint 1", etc.), the agent learns to generate programs that draw the corresponding MNIST digits. We can see that the generated drawings of a given label show diversity in style. (c) and (d): Results corresponding to (a) and (b) respectively for the fixed pixel-based reward function baseline, which also show little diversity in the generated outputs.

generator learns to generate the distribution of programs consistent with the symbolic instruction. In particular, this setting leads to more accurate and more diverse programs than hand-crafted discriminators such as L2-distance (in pixel space) between the generated and goal images.

Language conditioned policies with adversarial reward induction has also been explored in [1]. However, [1] do not evaluate whether the agent is capable of capturing the diversity in the programs. Moreover, we train our agents on more visually complex tasks involving 3D scenes.

Representing images or videos by latent programs has also shown promise in learning meaningful representations of data [21, 16, 11, 12, 11, 14, 20, 3, 1]. Such frameworks not only improve the interpretability of otherwise obscured decision making systems, but also have the potential to generalize better to unseen data, owing to stronger inductive biases. For instance, [11] use latent programs to infer novels poses of 3D objects, [20] show how to invert physical simulation which in turn can be used for inferring objects' dynamics, and [21] achieve state-of-the-art on the CLEVR visual question answering problem.

Contributions. We (1) extend the recently proposed reinforced adversarial learning framework [3] to instruction conditioned policies, (2) present the experimental results on two instruction-conditioned domains (drawing MNIST digits with a paint software and constructing scenes in a 3D editor), and (3) show that the proposed setting leads to more accurate and more diverse programs than a fixed pixel-based reward function baseline.

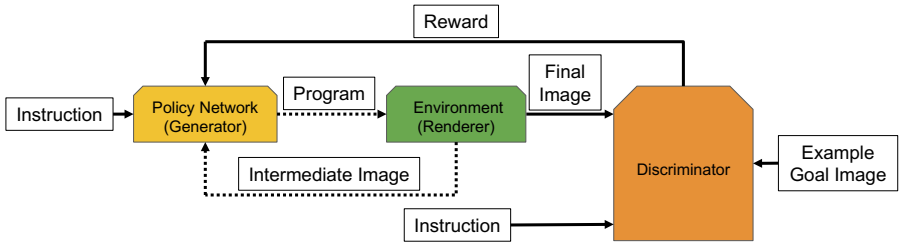

Figure 2: An overview of our approach. Given an instruction the generator (policy network) outputs a program which is rendered by a non-differentiable (denoted by dashed arrows) renderer into an image. This process repeats for a fixed number of time steps. The final image is fed to a discriminator, along with the instruction (the same that is given to the generator), that produces a reward.

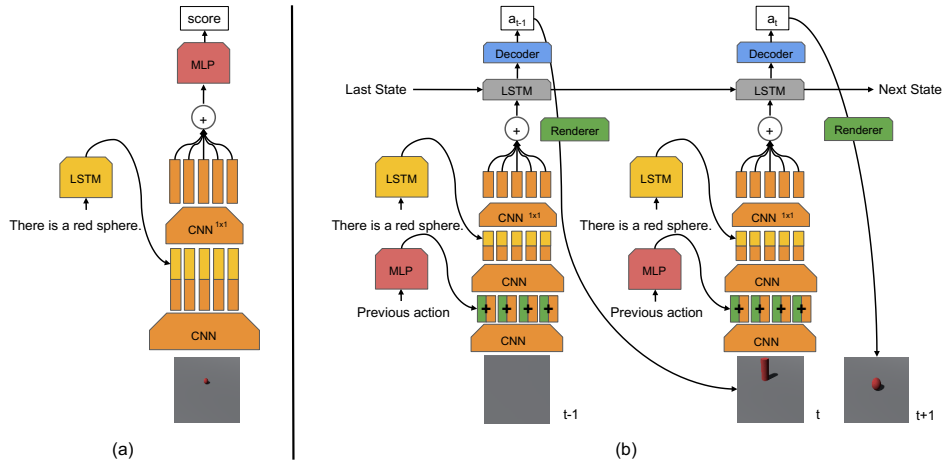

Figure 3: (a) Discriminator's architecture. It takes as inputs the instruction and an image (either the image generated by the agent or a goal image from the dataset), learns a joint embedding, and outputs a scalar score evaluating the 'realness' of the input pair. (b) Policy Network's architecture, unrolled over time. At each time step, it takes – an instruction, the rendered image from the previous time step (blank canvas initially) and the previous action, learns a joint embedding and passes it to an LSTM whose output is fed to a decoder which samples the actions.

2 Approach

Our goal is to train a generator agent G that is capable of generating programs for diverse scenes (x) conditioned on an instruction q that are indistinguishable from the ground truth data samples $p_d(x|q)$. Towards this goal, we frame our problem within the reinforcement adversarial learning setting [4, 15]. The generator is modeled by a recurrent policy network which, at every time step, takes a symbolic instruction as input, and outputs a program which is executed by a renderer to produce an image (Figure 2 shows the whole pipeline). For instance, if the instruction is "Draw zero", the generator uses the available painting tools (location, pressure, and brush size) to sequentially paint the digit 0 on the canvas. Similarly, with the input "There is a red sphere" the generator manipulates the scene by adding / changing objects so that the resulting scene has a red sphere. The role of the discriminator D is to decide if the distribution over generated images given an instruction (p_g) follows the ground truth distribution of images given the instruction ($p_g \approx p_d$).

Discriminator. To train the discriminator, we use LSTM [9] to transform the text instruction into a fixed-length vector representation. Using CNNs, we transform the input image into a spatial tensor. Inspired by architectures used in visual question answering [13] and text-to-image Generative Adversarial Networks [17], we concatenate each visual spatial vector with the replicated LSTM vector, and process both modalities by 4 layers of 1-by-1 CNNs. These representations are then summarized by a permutation-invariant sum-pooling operator followed by an MLP that produces a scalar score which is used as the reward for the generator agent (Figure 3a). We train the discriminator by mini-

mizing the minimax objective from [5]: $\mathcal{L}_D = -\mathbb{E}_{x\sim p_d}\left[\log\left(D(x|q)\right)\right] - \mathbb{E}_{x\sim p_g}\left[\log\left(1 - D(x|q)\right)\right] + R$, where R denotes gradient penalty, gently enforcing Lipschitz continuity [6].

Generator. Generator's goal is to generate programs that lead to scenes that match data distribution $p_d(x|q)$. Here, programs are defined as a sequence of commands $\mathbf{a} = (a_1, a_2, \ldots, a_N)$. Every command in this sequence is transformed into scenes through a domain-specific transformation $\mathcal{T}(\cdot)$. For instance, if $\mathcal{T}$ is a 3D renderer, and $\mathbf{a}$ is a program that describes a scene, $\mathcal{T}(\mathbf{a})$ is the image created by executing the program by the renderer. The loss function for the generator is set up adversarially to that for D, i.e. $\mathcal{L}_G = -\mathbb{E}_{x\sim p_g}\left[D(x)\right]$. Due to the non-differentiability of the renderer, we train the network with the REINFORCE [19] and advantage actor-critic (A2C), $\mathcal{L}_G = -\sum_t \log \pi(a_t|s_t;\theta)\left[R_t - V^\pi(s_t)\right]$, following [3]. Figure 3b depicts the architecture.

3 Experiments

MNIST digit painting. We constructed a dataset consisting of simple, templated instructions such as "Draw zero.", "Paint 5." etc. paired with random images for the corresponding MNIST digit. Each instruction follows the template: $< Action >< ClassLabel >$ with $Action := Draw \mid Put \mid Paint \mid Add \mid Create$ and $< ClassLabel >$ is specified either numerically ('0') or in word format ('zero'). This dataset has 60K image-instruction pairs (one instruction per image) in total. Following [3], we use the 'libmypaint' painting library as our rendering environment. The agent produces a sequence of strokes on a canvas C using a brush. Each action involves predicting 5 discrete quantities – end point and control point of the brush (on 32 x 32 grid), pressure applied to the brush (10 levels), brush size (4 levels) and a binary flag to choose between producing a stroke and skipping (not painting anything). So, the size of the action space is 83,886,080.

Figure 1b shows samples of MNIST digits painted by our agent when conditioned on the corresponding instructions. We can see that the agent draws the correct digit most of the time and for a given instruction, shows diversity in the generated samples, unlike the fixed pixel-based reward function baseline (L2) as shown in Figure 1d. Moreover, the baseline takes twice as much time as our main agent before it starts generating MNIST-like images.

Quantitatively, to evaluate the correctness (whether the generated samples are consistent with the instruction) and diversity (for a given instruction, whether the generated samples are diverse), we report Inception Score [18] (a combined score of quality and diversity) and Fréchet Inception Distance (FID) [8] (a measures of how close the generated image distribution is to the real image distribution) for 1000 samples using a pretrained MNIST classifier (Table 1). We can see that our learned Discriminator outperforms the fixed pixel-based reward baseline (L2).

	MNIST Digit Painting		3D Scene Construction		
Approach	Inception Score	FID	Correctness	Diversity in Colors	Diversity in Sizes
L2	1.22	283.5	1.4	0.77	0.09
Discriminator	**1.39**	**259.7**	**6.8**	**1.48**	**0.53**

Table 1: Quantitative evaluation of Discriminator and L2 (in pixel space). The former outperforms the latter on two domains that we use in our experiments. Depending on the domain, we report Inception Score, Correctness, Diversities – the higher the better, and FID – the lower the better.

3D scene construction. Inspired by CLEVR [10], we constructed a dataset consisting of images of 3D scenes and textual instructions. Each scene consists of a 3D object (cylinder, sphere, cube) of varying sizes, colors that are placed at different locations on the 32 x 32 grid. The instructions are generated using templates of the form "There is a $< Attribute >< Shape >$", where attributes are either colors (8 values) or sizes (2 values), and shapes are: cubes, spheres or cylinders. This dataset has 32,318 image-instruction pairs in total. We use a 3D editor as our rendering environment. The agent is provided with an empty scene and it adds objects sequentially. Each action involves predicting 5 discrete quantities – location of the object (on 32 x 32 grid), object shape, size and color and a flag to choose between adding a new object, changing the previously added object, or doing nothing. So, the size of the action space is 147,456.

Figure 1(a) shows samples of scenes generated by our agent when conditioned on the corresponding instructions. We can see that the generated scenes are consistent with the instruction and also show diversity in the attribute of the object which is not mentioned in the instruction (e.g., diversity in

colors when the instruction is 'There is a small sphere'.). The fixed pixel-based reward function baseline (L2) (Figure 1(c)), fails poorly in following the instruction.

Quantitatively, we evaluate the correctness and diversity for the unspecified attribute (for samples which are correct) as reported in Table 1 for 10 samples for each instruction. For measuring the diversity, we compute the entropy of the corresponding attribute distribution (number of correct samples for each of the 8 colors / 2 sizes). We can see that our learned Discriminator outperforms the L2 baseline.

Acknowledgments

The authors wish to acknowledge Igor Babuschkin, Caglar Gulcehre, Pushmeet Kohli, Piotr Trochim, and Antonio Garcia for their advice in this project.

References

[1] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. Learning to follow language instructions with adversarial reward induction. arXiv preprint arXiv:1806.01946, 2018.

[2] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[3] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In Proceedings of the International Conference on Machine Learning (ICML), 2018.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS), pp. 2672–2680, 2014.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In NIPS, 2014.

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In Advances in Neural Information Processing Systems (NIPS), pp. 5767–5777, 2017.

[7] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded language learning in a simulated 3d world. CoRR, abs/1706.06551, 2017.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. CoRR, abs/1706.08500, 2017.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[10] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pp. 1988–1997. IEEE, 2017.

[11] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4390–4399, 2015.

[12] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In Advances in Neural Information Processing Systems (NIPS), pp. 2539–2547, 2015.

[13] Mateusz Malinowski and Carl Doersch. The visual qa devil in the details: The impact of early fusion and batch norm on clevr. arXiv preprint arXiv:1809.04482, 2018.

[14] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In Advances in Neural Information Processing Systems (NIPS), pp. 1682–1690, 2014.

[15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pp. 1928–1937, 2016.

[16] Vinod Nair, Josh Susskind, and Geoffrey E Hinton. Analysis-by-synthesis by learning to invert generative black boxes. In International Conference on Artificial Neural Networks (ICANN), pp. 971–981. Springer, 2008.

[17] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. CoRR, abs/1605.05396, 2016.

[18] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. CoRR, abs/1606.03498, 2016.

[19] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4):229–256, 1992.

[20] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. In Advances in Neural Information Processing Systems (NIPS), pp. 153–164, 2017.

[21] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. arXiv preprint arXiv:1810.02338, 2018.