A. Level Generation and Environment Details

A.1. CoinRun

Each CoinRun level has a difficulty setting from 1 to 3. To generate a new level, we first uniformly sample over difficulties. Several choices throughout the level generation process are conditioned on this difficulty setting, including the number of sections in the level, the length and height of each section, and the frequency of obstacles. We find that conditioning on difficulty in this way creates a distribution of levels that forms a useful curriculum for the agent. For more efficient training, one could adjust the difficulty of sampled levels based on the agent's current skill, as done in (Justesen et al., 2018). However, we chose not to do so in our experiments, for the sake of simplicity.

At each timestep, the agent receives a $64 \times 64 \times 3$ RGB observation, centered on the agent. Given the game mechanics, an agent must know it's current velocity in order to act optimally. This requirement can be satisfied by using frame stacking or by using a recurrent model. Alternatively, we can include velocity information in each observation by painting two small squares in the upper left corner, corresponding to x and y velocity. In practice, agents can adequately learn under any of these conditions. Directly painting velocity information leads to the fastest learning, and we report results on CoinRun using this method. We noticed similar qualitative results using frame stacking or recurrent models, though with unsurprisingly diminished generalization performance.

A.2. CoinRun-Platforms

As with CoinRun, agents receive a $64 \times 64 \times 3$ RGB observation at each timestep in CoinRun-Platforms. We don't paint any velocity information into observations in experiments with CoinRun-Platforms; this information can be encoded by the LSTM used in these experiments. Unlike CoinRun, levels in CoinRun-Platforms have no explicit difficulty setting, so all levels are drawn from the same distribution. In practice, of course, some levels will still be drastically easier than others.

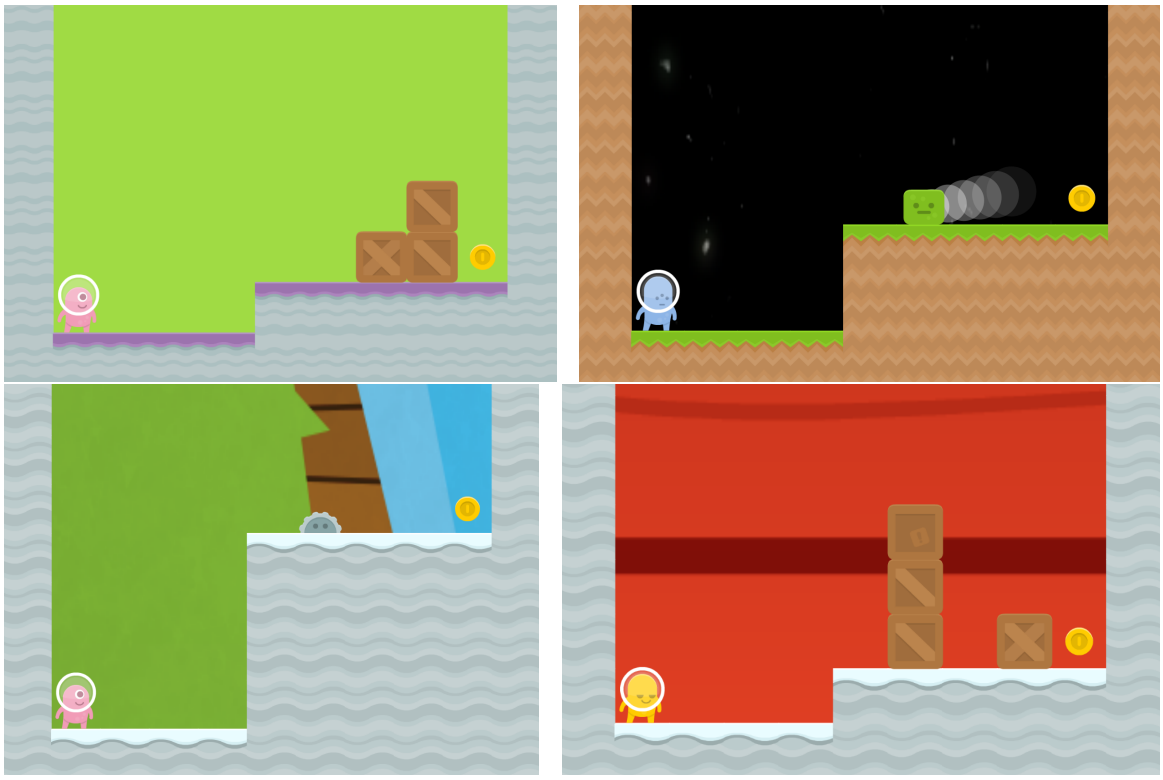
It is worth emphasizing that CoinRun platforms is a much more difficult game than CoinRun. We trained for 2B timesteps in Figure 7, but even this was not enough time for training to fully converge. We found that with unrestricted training levels, training converged after approximately 6B timesteps at a mean score of 20 per level. Although this agent occasionally makes mistakes, it appears to be reasonably near optimal performance. In particular, it demonstrates a robust ability to explore the level.

A.3. RandomMazes

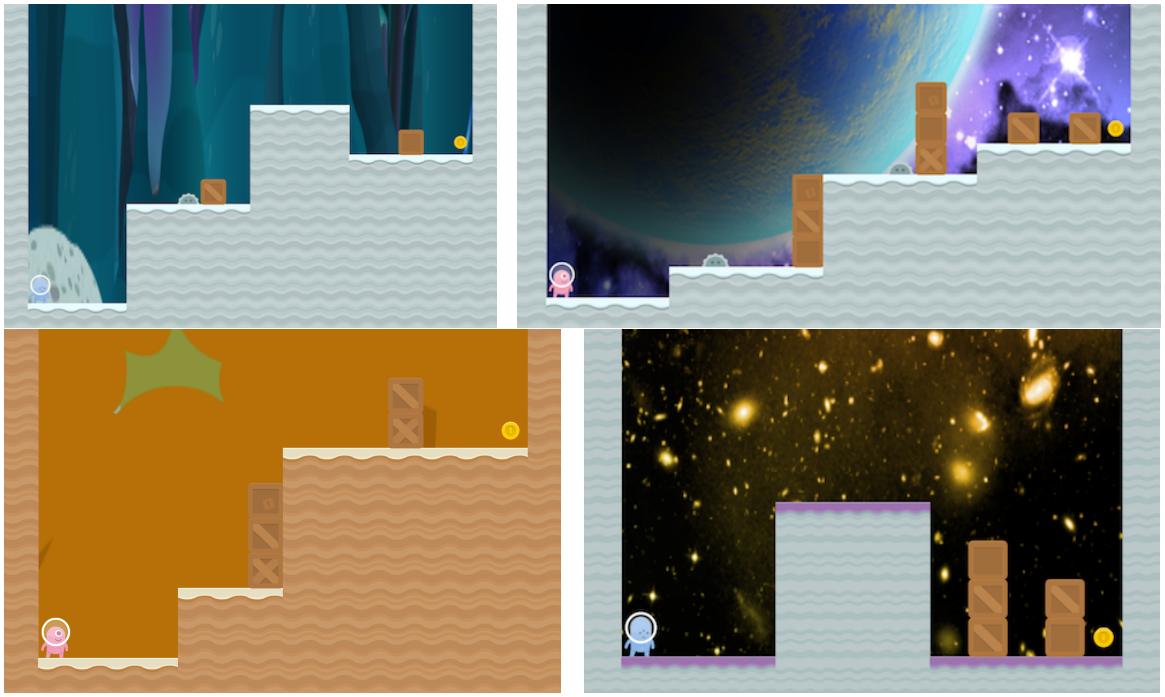
As with the previous environments, agents receive a $64 \times 64 \times 3$ RGB observation at each timestep in RandomMazes. Given the visual simplicity of the environment, using such a larger observation space is clearly not necessary. However, we chose to do so for the sake of consistency. We did conduct experiments with smaller observation spaces, but we noticed similar levels of overfitting.

B. Environment Screenshots

CoinRun Difficulty 1 Levels:

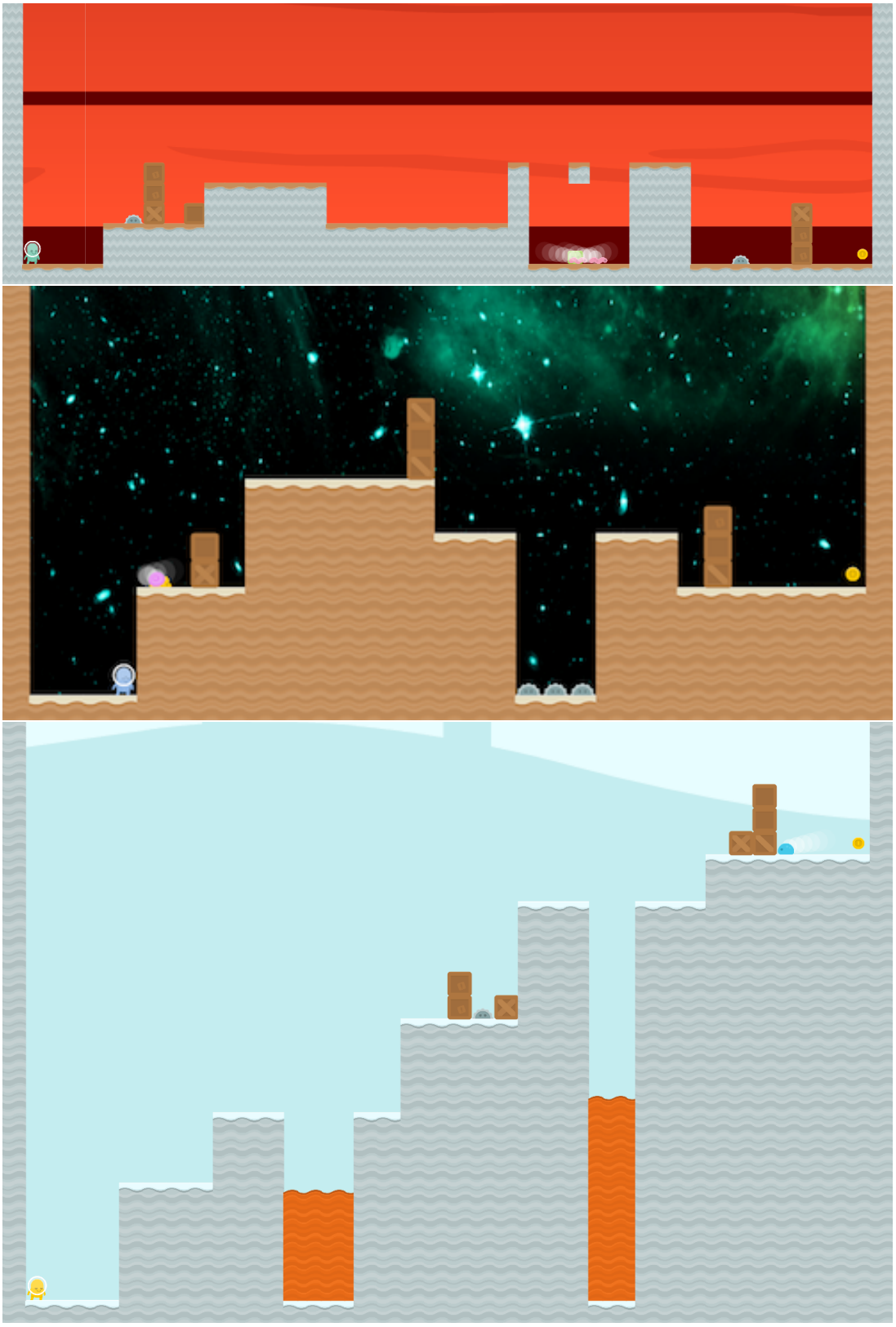


CoinRun Difficulty 2 Levels:



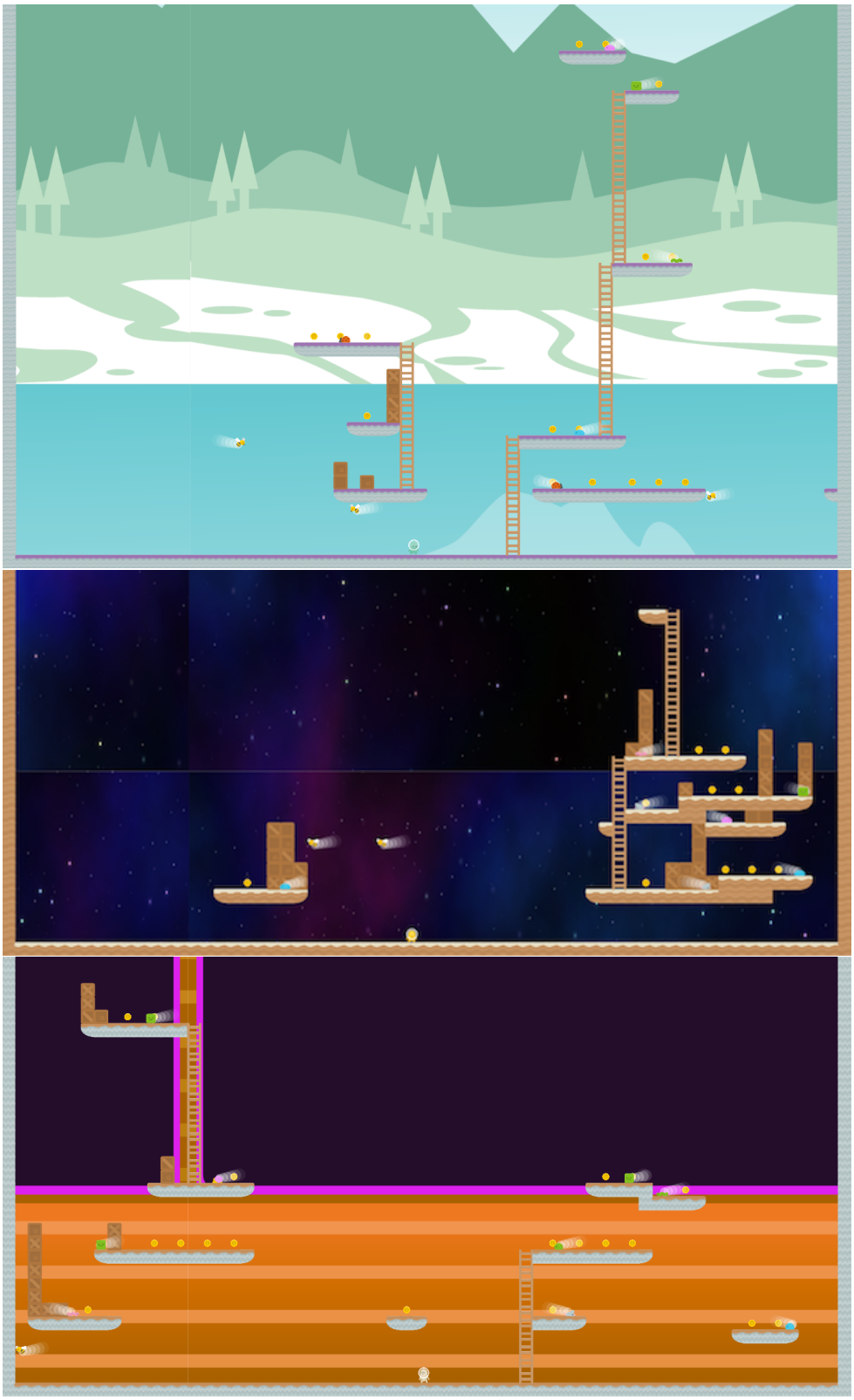
Note: Images scaled to fit.

CoinRun Difficulty 3 Levels:



Note: Images scaled to fit.

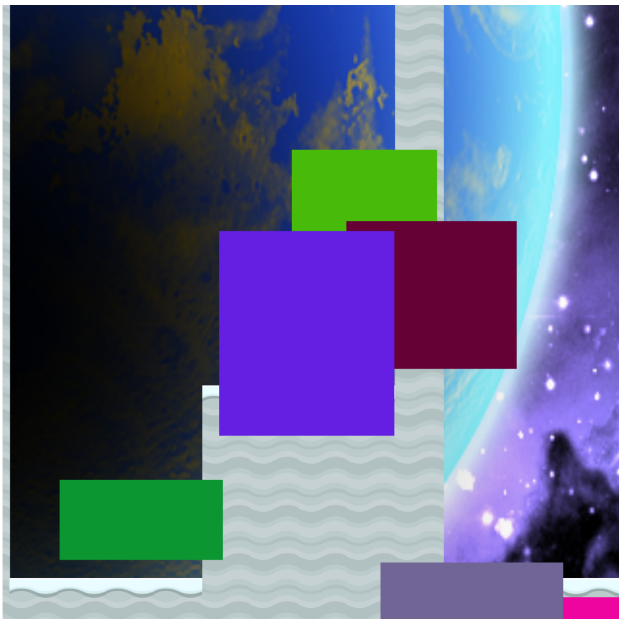
CoinRun-Platforms Levels:

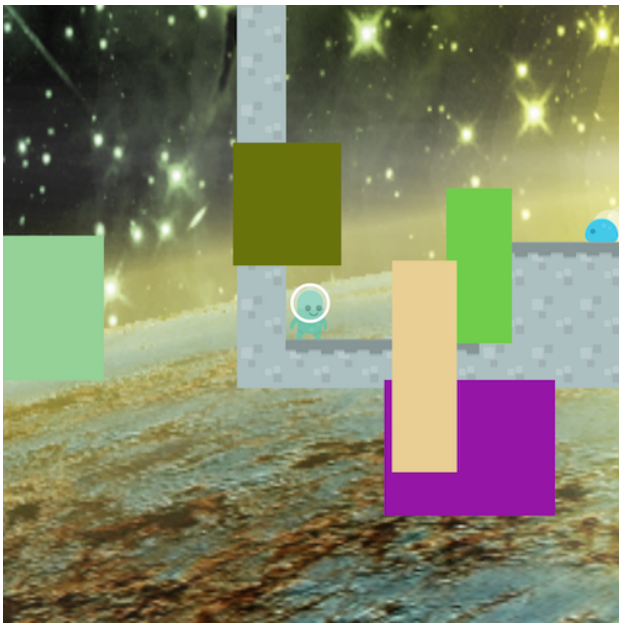


Note: Images scaled to fit.

C. Data Augmentation Screenshots

Example observations augmented with our modified version of Cutout (Devries and Taylor, 2017):





D. Hyperparameters and Settings

We used the following hyperparameters and settings in our baseline experiments with the 3 environments. Notably, we forgo the LSTM in CoinRun, and we use more environments per worker in CoinRun-Platforms.

	CoinRun	CoinRun-Platforms	RandomMazes
γ	.999	.999	.999
λ	.95	.95	.95
# timesteps per rollout	256	256	256
Epochs per rollout	3	3	3
# minibatches per epoch	8	8	8
Entropy bonus (k_H)	.01	.01	.01
Adam learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}
# environments per worker	32	96	32
# workers	8	8	8
LSTM?	No	Yes	Yes

E. Performance

# Levels	Nature Train	Nature Test	IMPALA Train	IMPALA Test
100	99.45 ± 0.09	66.79 ± 1.09	99.39 ± 0.08	66.58 ± 1.91
500	97.85 ± 0.46	70.54 ± 0.62	99.16 ± 0.19	80.25 ± 1.07
1000	95.7 ± 0.65	72.51 ± 0.68	97.71 ± 1.04	84.84 ± 2.24
2000	92.65 ± 0.71	75.6 ± 0.28	97.82 ± 0.32	90.92 ± 0.45
4000	90.18 ± 1.04	78.35 ± 1.47	97.7 ± 0.19	95.87 ± 0.62
8000	88.94 ± 1.08	84.02 ± 0.96	98.13 ± 0.21	97.29 ± 1.04
12000	89.11 ± 0.58	86.41 ± 0.46	98.14 ± 0.56	97.51 ± 0.46
16000	89.24 ± 0.77	87.58 ± 0.79	98.04 ± 0.17	97.77 ± 0.68
∞	90.87 ± 0.53	90.04 ± 0.9	98.11 ± 0.26	98.29 ± 0.16

Table 1. CoinRun (across 5 seeds)

# Levels	Train	Test
100	14.04 ± 0.86	2.22 ± 0.2
400	12.16 ± 0.11	5.74 ± 0.22
1600	10.36 ± 0.35	8.91 ± 0.28
6400	11.71 ± 0.73	11.38 ± 0.55
25600	12.23 ± 0.49	12.21 ± 0.64
102400	13.84 ± 0.82	13.75 ± 0.91
409600	15.15 ± 1.01	15.18 ± 0.98
∞	15.96 ± 0.37	15.89 ± 0.47

Table 2. CoinRun-Platforms IMPALA (across 3 seeds)

# Levels	Train	Test
1000	92.09 ± 0.35	68.13 ± 1.3
2000	93.57 ± 2.06	74.08 ± 1.25
4000	92.94 ± 1.47	77.79 ± 1.87
8000	98.47 ± 0.4	83.33 ± 0.66
16000	99.19 ± 0.19	87.49 ± 1.45
32000	98.62 ± 0.37	93.07 ± 1.0
64000	98.64 ± 0.14	97.71 ± 0.24
128000	99.06 ± 0.21	99.1 ± 0.34
256000	98.97 ± 0.18	99.21 ± 0.11
∞	98.83 ± 0.71	99.36 ± 0.16

Table 3. RandomMazes IMPALA (across 3 seeds)