Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Both the limitations and potential societal impacts are discussed in the appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The information is provided in the Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We use the MIT licence for our codebase, discussed in the Appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Design choices for reinforcement learning

Table A1: Design choices made in representation learning for reinforcement learning. Act, Aug, Mom, Proj and Comp respectively show whether action conditioning, augmentation, momentum, projection heads, and compression were used. P/J determines whether the representation learning is an initial (P)retraining step, or is (J)ointly learned alongside reinforcement learning. R/C/B/N in the Task column refer to Reconstruction, Contrastive, Bootstrap, or None. Note that different papers may use different sets of augmentations.

Algorithm	Task	RL alg.	Context	Target	Act	Aug	Mom	Proj	Comp	P/J
World models [16]	R	CMA-ES	o_t	o_t, o_{t+1}	✓	✗	✗	✗	✗	P
DVRL [30]	R	A2C	o_t	o_{t+k}	✓	✗	✗	✗	✗	J
PlaNet [17]	R	MPC + CEM	$o_{1:t}$	$o_{t+1:T}, r_{t+1:T}$	✓	✗	✗	✗	✗	J
SLAC [31]	R	SAC	o_t	o_{t+1}	✓	✗	✗	✗	✗	J
Poke [32]	R	-	o_t, o_{t+1}	a_t	-	✗	✗	✗	✗	-
RAD [4]	N	PPO, SAC	o_t	-	-	✓	-	-	✗	J
DrQ [5]	N	DQN, SAC	o_t	-	-	✓	-	-	✗	J
CURL [18]	C	DQN, SAC	o_t	o_t	✗	✓	✓	✗	✗	J
CPC [12]	C	A2C	o_t	o_{t+k}	✗	✗	✗	✗	✗	J
Bottleneck [33]	C	A2C	o_t	o_{t+k}	✓	✗	✗	✗	✗	J
DRIML [34]	C	C51	o_t	o_{t+k}	✓	✗	✗	✗	✗	J
PI-SAC [3]	C	SAC	o_t	o_{t+k}, r_{t+k}	✓	✓	✓	✓	✓	J
ATC [19]	C	SAC, PPO	o_t	o_{t+k}	✗	✓	✓	✓	✗	P
PBL [35]	B	PopArt-IMPALA	$o_{1:t}$	o_{t+k}	✓	✗	✗	✓	✗	J
SPR [36]	B	DQN	o_t	$o_{t+1:T}$	✓	✓	✓	✓	✗	J
M-CURL [37]	C	DQN, SAC	$o_{1:t}$	o_t	✗	✓	✓	✗	✗	J
PlayVirtual [38]	B	DQN, SAC	o_t	$o_{t+1:T}$	✓	✓	✓	✓	✗	J

B Additional information on experiment setup

B.1 Environment setup

DMC. The training set for each DMC task consist of 250 trajectories produced by an expert policy trained with RAD [4]. For each of our methods we report the mean return for the final policy, which ranges between 0 and 1,000.

Procgen. For each Procgen task, we reserve 100 environment seeds as "training levels", then use a separate 100 seeds as "testing levels" to evaluate generalization. The training set for each task consists of 110K frames produced by applying RAD's trained agent to the training levels. We report mean returns on the train and test levels separately in our results.

MAGICAL. We give the IL and RepL algorithms access to five human demonstrations from the demo variant, then provide RepL with an additional 150,000 time steps of random rollouts (1,250 to 3,750 trajectories, depending on the environment). This simulates situations where human expert demonstrations are expensive to collect, but robot exploration is relatively cheap. For space reasons, we report mean scores averaged across all variants, which range between 0 and 1; full results are in Appendix G.

B.2 RepL algorithms

As is discussed in the main text, we evaluated 5 RepL algorithms. SimCLR and TemporalCPC are contrastive baselines: SimCLR must assign similar representations to augmented copies of the same observation, and different representations to augmented copies of different observations. TemporalCPC must additionally account for dynamics by assigning similar representations to any pair of frames that are separated by a gap of Δt time steps (we use $\Delta t = 8$). Like SimCLR, the VAE attempts to represent one frame at a time, without using temporal offsets. We also include methods that explicitly condition on or generate actions: dynamics predicts the next observation given the

current observation and action, while inverse dynamics predicts which action was used to transition between two adjacent states. Note that we do not use momentum, projection heads, or compression in our final experiments, since our preliminary experiments did not show a significant advantage to doing so.

C Related benchmarking work in RL and IL

Table A2: Summary of previous works comparing different representation learning (repL) algorithms on imitation learning (IL) and reinforcement learning (RL). These works differ along a few axis on whether they provide a design breakdown of repL algorithms, their area of focus, the number of benchmarks covered, the number of algorithms experimented, whether they evaluate image-based environments, and whether they compare their results with an image augmentation baseline.

Research	Design Breakdown	Area	# of Benchmarks	# of Algo.	Image env.	Aug
ACL [26]	✓	RL, IL	1	10	✗	✗
DrQ [5]	✗	RL	1	6	✓	✓
RAD [4]	✗	RL	3	8	✓	✓
EIRLI (ours)	✓	IL	3	7	✓	✓

To the best of our knowledge, there are three existing work that survey RepL methods on imitation learning and reinforcement learning. We summarize their differences in Table A2. ACL [26] provided a design breakdown of different RepL algorithms too, and they found that many RepL algorithms perform poorly on imitation learning but can provide extra benefits for offline RL. DrQ [5] and RAD [4] both discussed the effect of image augmentation in reinforcement learning in great detail, and showed that with well-tuned image augmentations, a standard reinforcement learning framework can outperform many self-supervised learning methods on RL.

D Hyperparameter details

Hyperparameter	Value	Tuning range
All algorithms		
Optimizer	Adam	-
LR	10^{-4}	10^{-6}–10^{-2}
Training batches	5,000	-
Representation dim.	128	64–256
VAE, dyn., inv. dyn.		
Batch size	64	-
Augmentations	-	-
VAE		
VAE β	10^{-6}	10^{-7}–1.0
SimCLR, TCPC		
Batch size	384	64–512
Augmentations	trans. rot., blur, col. jit.	-
TCPC		
Temporal offset	8 steps	-

Table A3: Hyperparameters for representation learning. Note that for joint training, BC and RepL use the same optimizer, and thus have the same learning rate.

Hyperparameter	Value	Tuning range
All benchmarks		
Optimizer	Adam	-
LR	10^{-4}	-
Entropy coeff.	10^{-3}	-
ℓ_2 reg. coeff.	10^{-5}	-
Augmentations	trans., rot., blur, col. jit.	-
All benchmarks, pretraining		
Batch size	32	-
All benchmarks, joint training		
Batch size	64	-
dm_control and Procgen, pretraining and joint training		
Training batches	1M	1M–4M
MAGICAL, pretraining		
Training batches	20k	5k–20k
MAGICAL, joint training		
Training batches	30k	-

Table A4: Hyperparameters for behavioral cloning. Sections marked "pretraining" show the hyperparameters used for BC *after* pretraining; sections marked "joint training" apply to BC during joint training. Representation learning hyperparameters, such as the batch size, are covered separately in Table A3.

Environments and datasets For each dm_control environment, we generated synthetic demonstration data using RAD with default algorithm hyperparameters [4].[2] Environment configurations (such as action repeat, frame stack, etc.) were the same for both RAD and our IL algorithms. Specifically:

- In cheetah-run, we used an action repeat of 4, resulting in a trajectory length of $1000/4 = 250$. Our demonstration dataset consisted of 250 trajectories (62,500 time steps) from the RAD demonstration agent, with a mean return of ≈827 (recall that return ranges between 0 and 1,000 for all DMC environments).
- In finger-spin, we used an action repeat of 2, resulting in a trajectory length of $1000/2 = 500$. Our dataset again consisted of 250 trajectories (125,000 time steps) sampled from the RAD demonstration agent, with mean return of ≈963.
- In reacher-easy, we used an action repeat of 8, resulting in a trajectory length of $1000/8 = 125$. Our dataset of 250 trajectories (31,250 time steps) had mean return ≈977, and was again generated by RAD.

For all DMC environments, we used a frame stack of 3.

As with DMC, we generated expert demonstrations for Procgen using a policy trained with RAD.[3] We used the `easy` variants of all environments, with a frame stack of 3 and no action repeat. We used a demonstration dataset of around 114,000 timesteps for each agent. The mean trajectory lengths and returns are as follows:

- For CoinRun, trajectories averaged 26 steps, and the demonstrator had an average return of 8.7.
- For Fruitbot, trajectories had an average length of 442, and the demonstrator attained a mean return of 29.75.
- For Jumper, trajectories had an average length of 76, and mean return of 8.7.

[2] `https://github.com/MishaLaskin/rad`

[3] `https://github.com/pokaxpoka/rad_procgen`

Hyperparameter	DMC	Procgen	MAGICAL	Tuning range
Policy (PPO)				
# parallel envs	32	32	32	-
Time steps per round	8	10	7	4–12
Epochs per round	12	9	7	4–12
Adam minibatch size	48	48	48	-
Initial Adam step size	10^{-4}	10^{-4}	2.5×10^{-4}	$5 \times (10^{-5}$–$10^{-4})$
Final Adam step size	Linearly annealed to 0 over training			-
Discount γ	0.99	0.6	0.99	0.6–1
GAE λ	0.8	0.6	0.76	0.6–0.9
Entropy bonus	10^{-8}	5×10^{-6}	4.5×10^{-8}	10^{-10}–10^{-3}
Advantage clip ϵ	0.02	0.01	0.006	0.001–0.1
Grad. clip ℓ_2 norm	1	1	1	-
Augmentations	-	-	-	-
Discriminator				
Batch size	48	48	48	-
Adam step size	10^{-3}	2.5×10^{-3}	5.7×10^{-4}	$5 \times (10^{-4}$–$10^{-3})$
Disc. steps per round	6	2	2	1–8
Augmentations	Erase, blur, noise, rot.	Col. jit., flip LR, blur, noise, rot., trans.	Col. jit., erase, flip LR, blur, noise, rot.	Col. jit., erase, flip LR, blur, noise, rot., trans.
Misc.				
Total env. steps of training	5×10^5	5×10^5	5×10^5	-
Reward norm. std. dev.	0.01	0.01	0.01	-

Table A5: Hyperparameters for GAIL experiments. We use the word "round" to describe the repeated sequence of data collection, followed by PPO updates on the collected data, followed by discriminator updates on both demonstrations and rollouts. Representation learning hyperparameters, such as the batch size, are covered separately in Table A3.

For each MAGICAL environment, we used a fixed subset of five demonstration trajectories (initially selected at random) from the human dataset provided with the benchmark [6]. We used egocentric views with a frame stack of four and no action repeat. Because there was no action repeat, trajectory lengths remained at the defaults for the benchmark suite: 40 for MoveToRegion, 80 for MoveToCorner, 120 for MatchRegions. For each benchmark, we used between 25 and 28 demonstration trajectories, and the demonstrator attained the maximum return of 1.0 (on a 0.0–1.0 scale) in each trajectory. In addition to demonstrations, our MAGICAL experiments also used random rollout datasets of 150,000 timesteps, all generated by uniformly sampling from the action set at each time step. This equates to between 1,250 and 3,750 trajectories, depending on the horizon of the task.

RepL hyperparameters Representation learning hyperparameters are given in Table A3. Note that the contrastive algorithms have slightly different hyperparameters from the other RepL algorithms. We found that a batch size close to 400 was important for contrastive algorithm performance; setting this value too low *or* too high (e.g. 500+) decreased performance. Predictive and reconstructive algorithms are less sensitive to batch size, so we used a computationally convenient batch size (64).

For the contrastive algorithms, we used a mixture of translation (trans.), rotation (rot.), Gaussian blur (blur), and color jitter (col. jit.) augmentations. The translation augmentation translates the image by up to 5% of image dimensions; the rotation augmentation rotates the image by up to 5°; the Gaussian blur augmentation applies a Gaussian blur kernel with $\sigma = 1$px; and the color jitter augmentation randomizes the hue by up to 0.15 radians. We did not find the algorithms were highly sensitive to the choice of augmentations, but these augmentations did appear to perform fractionally better than the other choices that we considered during manual tuning. For non-contrastive algorithms, we did not use augmentations.

For the VAE, we used a mean squared error loss for reconstruction, and down-weighted the KL prior term by a factor of β. Specifically, our loss was

$$\mathcal{L}_{\text{VAE}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 - \beta\,\mathrm{KL}(e_\theta(\cdot \mid x_i)\|\mathcal{N}(\cdot; 0, I)) \;,$$

where i indexes over the n elements of the output image. This is equivalent to using a traditional β-VAE with a fixed variance of $\sigma^2 = \frac{n}{2}$ for the Gaussian output distribution.

IL hyperparameters Imitation learning hyperparameters are given in Table A4 for BC, and Table A5 for GAIL. IL hyperparameters were the same for both control and RepL runs, except for the network initialization, where RepL runs initialized from the RepL-trained encoder, while control runs used a fresh He initialization [39]. We found that DMC and Procgen needed substantially more training than MAGICAL; we found that MAGICAL performance was plateauing even with an order of magnitude less training.

Compute information Running one experiment (a single seed of combined RepL and IL) on DMC and Procgen for 1M batch updates takes about 40 hours on one NVIDIA 1080Ti, and running one for MAGICAL between 30 minutes (for 20,000-batch control without augmentations) and 10 hours (for a contrastive method using joint training for 30,000 batches). Generating the results in this paper takes around 6,600 GPU hours for DMC, 8,800 GPU hours for Procgen, and 26–525 hours for MAGICAL (assuming 4 seeds per GPU, and 30 minutes to 10 hours per seed).

E Limitations, social impacts, and benchmark license

Limitations The main limitation of our findings is that we only investigate policy learning with BC. Our findings therefore may not generalise to IL algorithms that learn more than just a policy. This includes IRL algorithms, which typically learn both a reward function and a policy, as well as IL algorithms like SQIL [28] that learn a Q function rather than directly learning a policy.

Social impacts We do not foresee any negative near-term social impact from our work.

License We release our codebase and associated data under the MIT license.

F Implementation of components in the codebase

In Section 2 we analyzed several design axes and their components. We elaborate in this section our current implementation status of these components in the codebase.

Target selection. Different versions of this design choice are implemented by inheriting from the `TargetPairConstructor` class within the codebase. Currently implemented are `IdentityPairConstructor`, in which the context and target are identical, and `TemporalOffsetPairConstructor`, which can be given a desired temporal offset, and configured to optionally return the action as extra context.

Loss type. Different versions of loss functions are implemented by inheriting from `RepresentationLoss`. We have already implemented a wide variety of losses, including VAE, mean squared error, negative log likelihood, CEB, and several contrastive losses.

Augmentation. This design choice is implemented in subclasses of `Augmenter`. We rely on a standard library to implement the augmentations. Each subclass augments a different set of inputs: both the context and target, only the context, or neither the context nor target.

Encoder. Different versions of encoder are implemented by inheriting from the `Encoder` class within the codebase. We have implemented encoders that work on individual images, as well as a `RecurrentEncoder`.

Decoder. Different versions of decoder are implemented by inheriting from the `LossDecoder` class within the codebase. Currently implemented decoders support image reconstruction, action conditioning, and projection heads.

G Complete MAGICAL results

Task	Dynamics	InvDyn	SimCLR	TemporalCPC	VAE	BC, augs	BC, no augs
MatchRegions-Demo	0.71±0.06	0.72±0.07	0.72±0.06	0.68±0.03	0.70±0.05	0.76±0.05	0.63±0.15
-TestDynamics	0.55±0.09	0.56±0.07	0.56±0.07	0.52±0.01	0.56±0.08	0.66±0.05	0.38±0.12
-TestColour	0.71±0.06*	0.70±0.08*	0.67±0.04*	0.66±0.04*	0.67±0.06*	0.53±0.04	0.30±0.08
-TestShape	0.61±0.06	0.64±0.09	0.62±0.06	0.61±0.04	0.60±0.05	0.70±0.03	0.43±0.13
-TestJitter	0.70±0.06	0.65±0.06	0.66±0.07	0.65±0.04	0.68±0.03	0.68±0.05	0.41±0.13
-TestLayout	0.04±0.01	0.04±0.01	0.04±0.01	0.05±0.01	0.04±0.01	0.05±0.01	0.05±0.03
-TestCountPlus	0.04±0.02	0.04±0.01	0.06±0.03	0.05±0.01	0.04±0.02	0.05±0.02	0.04±0.02
-TestAll	0.04±0.02	0.05±0.02	0.05±0.03	0.06±0.01*	0.04±0.02	0.05±0.01	0.03±0.02
Average	0.42±0.04	0.42±0.04	0.42±0.03	0.41±0.01	0.42±0.03	0.43±0.02	0.28±0.08
MoveToCorner-Demo	0.94±0.08	0.92±0.07	0.88±0.09	0.86±0.10	0.89±0.08	0.86±0.08	0.99±0.01*
-TestDynamics	0.83±0.05	0.87±0.05*	0.85±0.04*	0.80±0.06	0.75±0.04	0.76±0.08	0.68±0.04
-TestColour	0.90±0.10	0.87±0.06	0.86±0.13	0.86±0.06	0.87±0.07	0.75±0.16	0.85±0.07
-TestShape	0.90±0.06	0.88±0.07	0.91±0.05*	0.87±0.02	0.84±0.09	0.84±0.06	0.71±0.10
-TestJitter	0.81±0.09	0.80±0.04	0.84±0.04*	0.79±0.04	0.77±0.04	0.78±0.00	0.58±0.08
-TestAll	0.66±0.13	0.64±0.09	0.65±0.07	0.62±0.06	0.57±0.12	0.67±0.09	0.51±0.13
Average	0.84±0.07	0.83±0.04	0.83±0.04*	0.80±0.02	0.78±0.06	0.78±0.05	0.72±0.04
MoveToRegion-Demo	0.99±0.01	0.99±0.01	1.00±0.01	0.98±0.02	0.99±0.02	1.00±0.00	1.00±0.00
-TestDynamics	0.99±0.01	1.00±0.01	0.99±0.01	0.99±0.01	0.99±0.01	1.00±0.01	1.00±0.00
-TestColour	0.99±0.02*	0.99±0.01*	1.00±0.00*	0.99±0.01*	0.97±0.04*	0.54±0.08	0.84±0.13*
-TestJitter	0.99±0.01	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.01	1.00±0.00	0.99±0.01
-TestLayout	0.52±0.05	0.51±0.05	0.45±0.03	0.44±0.07	0.46±0.11	0.64±0.03	0.64±0.11
-TestAll	0.45±0.06*	0.50±0.09*	0.46±0.06*	0.42±0.08*	0.47±0.12*	0.28±0.06	0.38±0.08*
Average	0.82±0.02*	0.83±0.02*	0.82±0.01*	0.81±0.01*	0.81±0.05*	0.74±0.02	0.81±0.04*

Table A6: Complete results for all variants of the evaluated MAGICAL tasks, with BC plus RepL pretraining. Refer to Table A7 for joint training.

Task	Dynamics	InvDyn	SimCLR	TemporalCPC	VAE	BC, augs	BC, no augs
MatchRegions-Demo	0.78±0.03	0.51±0.16	0.73±0.06	0.01±0.01	0.72±0.09	0.78±0.06	0.70±0.04
-TestDynamics	0.62±0.04	0.39±0.14	0.59±0.04	0.01±0.01	0.58±0.05	0.63±0.05	0.40±0.03
-TestColour	0.60±0.03*	0.17±0.07	0.55±0.06*	0.01±0.01	0.48±0.04	0.47±0.08	0.35±0.10
-TestShape	0.71±0.04*	0.36±0.14	0.67±0.05	0.01±0.01	0.68±0.05	0.66±0.04	0.48±0.02
-TestJitter	0.69±0.05	0.36±0.14	0.66±0.04	0.01±0.01	0.65±0.04	0.66±0.06	0.40±0.10
-TestLayout	0.05±0.01	0.04±0.01	0.04±0.01	0.01±0.01	0.05±0.02	0.07±0.01	0.03±0.01
-TestCountPlus	0.04±0.01	0.02±0.01	0.04±0.01	0.01±0.01	0.05±0.03	0.07±0.02	0.04±0.01
-TestAll	0.04±0.01	0.02±0.01	0.03±0.01	0.01±0.01	0.04±0.01	0.07±0.02	0.04±0.03
Average	0.44±0.02	0.23±0.08	0.41±0.02	0.01±0.01	0.41±0.03	0.43±0.03	0.31±0.02
MoveToCorner-Demo	0.86±0.04	0.37±0.25	0.90±0.12	0.03±0.03	0.94±0.05	0.95±0.04	0.99±0.00*
-TestDynamics	0.82±0.07	0.35±0.27	0.86±0.05	0.03±0.03	0.87±0.04	0.82±0.06	0.75±0.09
-TestColour	0.84±0.08	0.25±0.29	0.61±0.17	0.01±0.01	0.73±0.15	0.77±0.15	0.76±0.17
-TestShape	0.84±0.12	0.44±0.27	0.88±0.09	0.02±0.03	0.93±0.05	0.89±0.04	0.76±0.09
-TestJitter	0.76±0.06	0.25±0.18	0.78±0.06	0.02±0.03	0.83±0.07	0.81±0.08	0.55±0.12
-TestAll	0.57±0.11	0.13±0.15	0.53±0.10	0.01±0.01	0.63±0.09	0.57±0.10	0.39±0.21
Average	0.78±0.07	0.30±0.22	0.76±0.05	0.02±0.02	0.82±0.06	0.80±0.05	0.70±0.09
MoveToRegion-Demo	1.00±0.00	0.60±0.34	1.00±0.00	0.81±0.14	1.00±0.00	1.00±0.00	1.00±0.00
-TestDynamics	1.00±0.00	0.55±0.37	1.00±0.00	0.81±0.14	1.00±0.00	1.00±0.00	0.99±0.01
-TestColour	0.64±0.15	0.22±0.15	0.93±0.07*	0.25±0.06	0.69±0.12	0.65±0.13	0.77±0.12
-TestJitter	1.00±0.00	0.49±0.40	1.00±0.00	0.78±0.08	1.00±0.00	1.00±0.00	0.98±0.01
-TestLayout	0.64±0.05	0.17±0.16	0.30±0.06	0.10±0.05	0.68±0.06*	0.61±0.04	0.63±0.06
-TestAll	0.29±0.06	0.10±0.08	0.23±0.04	0.06±0.03	0.28±0.03*	0.23±0.03	0.31±0.08
Average	0.76±0.02	0.35±0.24	0.74±0.01	0.47±0.07	0.77±0.02	0.75±0.02	0.78±0.04

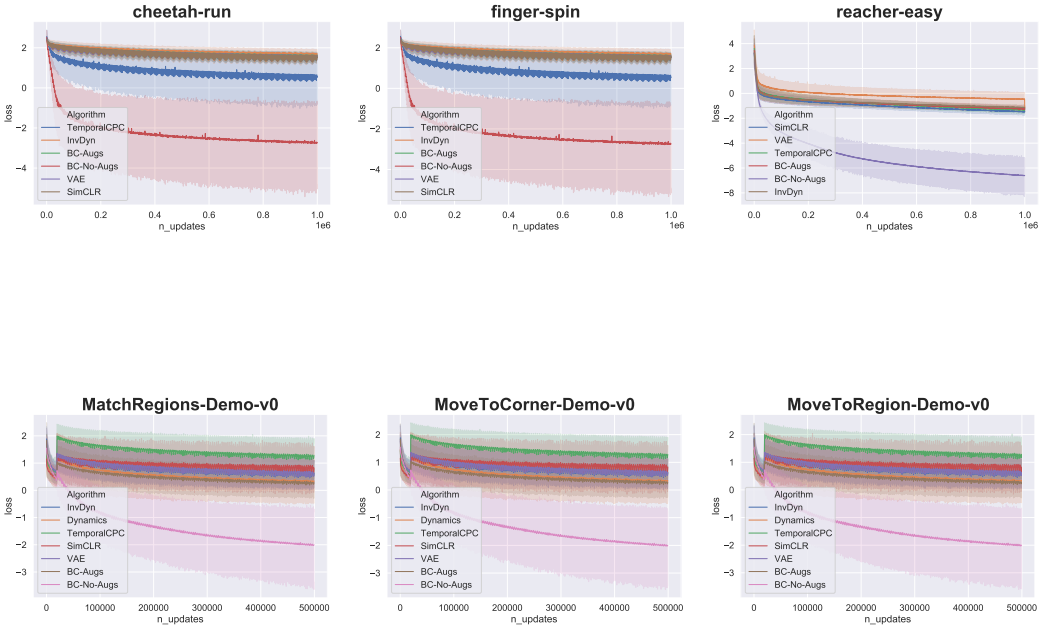
Table A7: Complete results for all variants of the evaluated MAGICAL tasks, with BC plus RepL joint training. Refer to Table A6 for BC plus RepL pretraining.

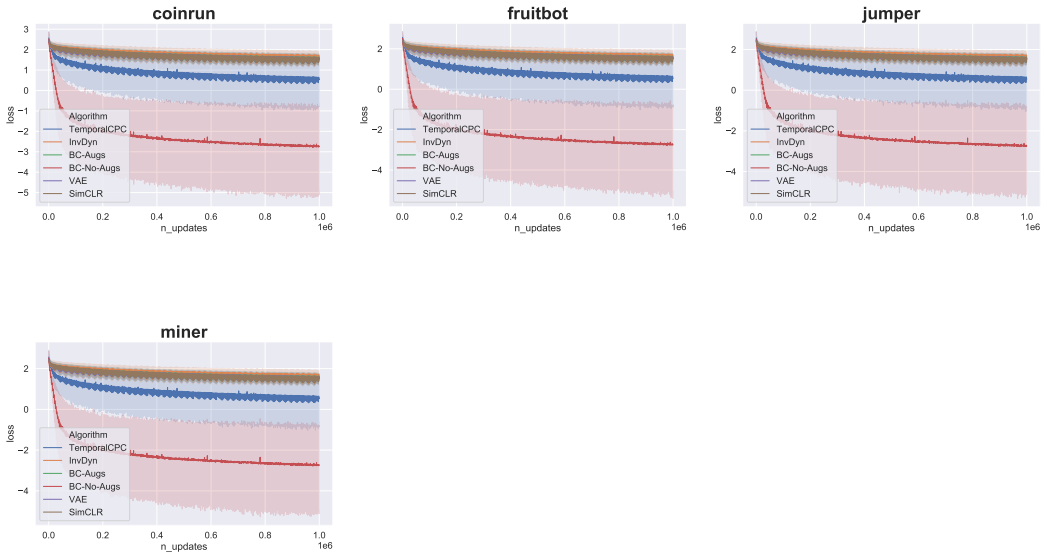
Task	Dynamics	InvDyn	SimCLR	TemporalCPC	VAE	GAIL augs	GAIL no augs
MatchRegions-Demo	0.76±0.19	0.59±0.24	0.83±0.09	0.68±0.22	0.51±0.28	0.81±0.10	0.36±0.22
-TestDynamics	0.74±0.17	0.55±0.22	0.79±0.11	0.64±0.25	0.51±0.27	0.77±0.13	0.37±0.22
-TestColour	0.21±0.04	0.18±0.06	0.25±0.04	0.21±0.07	0.14±0.08	0.24±0.03	0.08±0.05
-TestShape	0.72±0.20	0.56±0.22	0.81±0.10	0.61±0.25	0.49±0.26	0.76±0.10	0.35±0.20
-TestJitter	0.73±0.17	0.57±0.20	0.83±0.10	0.67±0.22	0.53±0.28	0.79±0.13	0.38±0.22
-TestLayout	0.11±0.03	0.13±0.04	0.11±0.03	0.15±0.06	0.13±0.06	0.17±0.04	0.12±0.06
-TestCountPlus	0.06±0.03	0.07±0.03	0.09±0.05	0.08±0.04	0.05±0.02	0.08±0.04	0.05±0.02
-TestAll	0.05±0.01	0.06±0.03	0.08±0.02	0.08±0.02	0.06±0.02	0.09±0.04	0.06±0.03
Average	0.42±0.10	0.34±0.12	0.47±0.04	0.39±0.12	0.30±0.15	0.46±0.06	0.22±0.12
MoveToCorner-Demo	0.67±0.09	0.60±0.09	0.67±0.14	0.72±0.15	0.78±0.07*	0.63±0.13	0.62±0.14
-TestDynamics	0.59±0.10	0.58±0.10	0.66±0.09	0.69±0.11	0.76±0.09*	0.65±0.12	0.64±0.16
-TestColour	0.35±0.08	0.32±0.16	0.39±0.09	0.45±0.21	0.51±0.19	0.37±0.10	0.57±0.16*
-TestShape	0.43±0.10	0.43±0.16	0.51±0.09	0.52±0.16	0.54±0.14	0.49±0.12	0.47±0.15
-TestJitter	0.61±0.10	0.54±0.09	0.62±0.12	0.63±0.16	0.74±0.08*	0.62±0.11	0.61±0.16
-TestAll	0.19±0.09	0.21±0.20	0.25±0.14	0.31±0.19	0.38±0.18*	0.20±0.13	0.39±0.16*
Average	0.48±0.09	0.45±0.10	0.52±0.07	0.55±0.15	0.62±0.11*	0.49±0.08	0.55±0.14
MoveToRegion-Demo	0.96±0.04	0.95±0.04	0.95±0.07	0.97±0.03	0.96±0.04	0.96±0.08	0.89±0.09
-TestDynamics	0.94±0.05	0.94±0.05	0.94±0.07	0.97±0.03	0.95±0.04	0.95±0.06	0.82±0.20
-TestColour	0.65±0.16	0.68±0.10	0.69±0.23	0.68±0.12	0.68±0.21	0.67±0.19	0.57±0.21
-TestJitter	0.92±0.06	0.93±0.05	0.96±0.08	0.97±0.04	0.97±0.03	0.94±0.08	0.69±0.25
-TestLayout	0.59±0.09	0.61±0.10	0.59±0.09	0.66±0.06	0.64±0.14	0.65±0.16	0.39±0.18
-TestAll	0.28±0.09	0.36±0.09	0.28±0.08	0.29±0.06	0.32±0.08	0.35±0.10	0.24±0.09
Average	0.72±0.07	0.74±0.04	0.74±0.06	0.76±0.03	0.75±0.07	0.75±0.09	0.60±0.14

Table A8: Complete results for all variants of the evaluated MAGICAL tasks, with GAIL plus RepL pretraining.

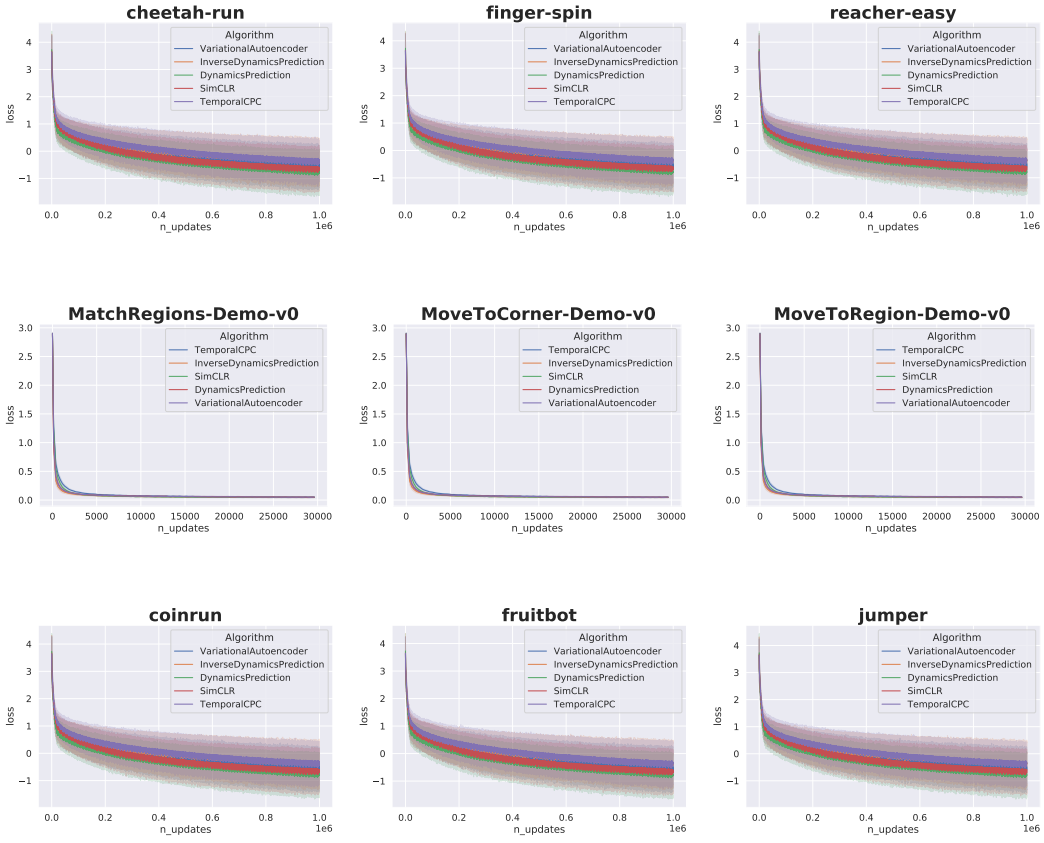
H Loss curves

H.1 Pretrain



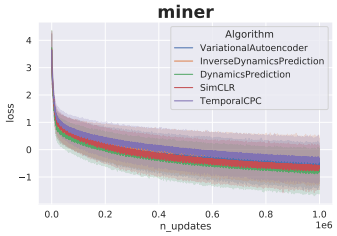


H.2 Joint Training



I Additional contrastive learning ablations

Table A9 presents ablations for SimCLR on our three MAGICAL tasks, using BC as the downstream IL algorithm. In particular, we experiment with:



Projection heads By default, SimCLR uses a "symmetric" projection strategy that applies the same projection head to the encoded contexts and encoded target before computing the loss. We also try using asymmetric projection heads, which are allowed to apply different transforms to the target and context embeddings, and additionally experiment with removing projection heads entirely so that we are computing SimCLR loss directly on the encoder representation.

Compression We experiment with compression by replacing the default SimCLR loss with the CEB loss [40], but leaving all other training and architecture details the same.

Momentum In the momentum ablation, we replace the SimCLR loss and encoder with a MoCo-style [13] loss and momentum encoder. Again, all other training and architecture details are left the same as in our SimCLR implementation.

In Table A9, we see that none of these modifications significantly improve performance over standard SimCLR. For this reason, we expect that using different projection heads, using compression, and using momentum are unlikely to affect the conclusions of our work.

Task	Asymm. proj.	No proj.	CEB loss	Momentum	SimCLR
MatchRegions-Demo	0.43±0.03	0.44±0.02	0.42±0.03	0.45±0.04	0.42±0.04
MoveToCorner-Demo	0.78±0.03	0.83±0.03	0.80±0.03	0.83±0.03	0.86±0.06
MoveToRegion-Demo	0.82±0.01	0.80±0.03	0.83±0.02	0.81±0.01	0.81±0.02

Table A9: Ablations for SimCLR variants on MAGICAL. We used SimCLR as a pretraining step for BC. Significance levels were evaluated relative to vanilla SimCLR (the rightmost column) using a one-sided Welch's t-test at $p < 0.05$, as with our other results. None of these results differ significnatly from SimCLR, and so none are starred.