

Appendix:

Unsupervised Reinforcement Learning Benchmark

A Unsupervised Reinforcement Learning Baselines

A.1 Knowledge-based Baselines

Prediction methods train a forward dynamics model $f(\mathbf{o}_{t+1}|\mathbf{o}_t, \mathbf{a}_t)$ and define a self-supervised task based on the outputs of the model prediction.

Curiosity [48]: The Intrinsic Curiosity Module (ICM) defines the self-supervised task as the error between the state prediction of a learned dynamics model $\hat{\mathbf{z}}' \sim g(\mathbf{z}'|\mathbf{z}, \mathbf{a})$ and the observation. The intuition is that parts of the state space that are hard to predict are good to explore because they were likely to be unseen before. An issue with Curiosity is that it is susceptible to the *noisy TV problem* wherein stochastic elements of the environment will always cause high prediction error while not being informative for exploration:

$$r_t^{\text{ICM}} \propto \|g(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) - \mathbf{z}_{t+1}\|^2.$$

Disagreement [49]: Disagreement is similar to ICM but instead trains an ensemble of forward models and defines the intrinsic reward as the variance (or disagreement) among the models. Disagreement has the favorable property of not being susceptible to the noisy TV problem, since high stochasticity in the environment will result high prediction error but low variance if it has been thoroughly explored:

$$r_t^{\text{Disagreement}} \propto \text{Var}\{g_i(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)\} \quad i = 1, \dots, N.$$

RND [10]: Random Network Distillation (RND) defines the self-supervised task by predicting the output of a frozen randomly initialized neural network \tilde{f} . This differs from ICM only in that instead of predicting the next state, which is effectively an environment-defined function, it tries to predict the vector output of a randomly defined function. Similar to ICM, RND can suffer from the noisy TV problem:

$$r_t^{\text{RND}} \propto \|g(\mathbf{z}_t, \mathbf{a}_t) - \tilde{g}(\mathbf{z}_t, \mathbf{a}_t)\|_2^2.$$

A.2 Data-based Baselines

Recently, exploration through state entropy maximization has resulted in simple yet effective algorithms for unsupervised pre-training. We implement two leading variants of this approach for URLB.

APT [42]: Active Pre-training (APT) utilizes a particle-based estimator [60] that uses K nearest-neighbors to estimate entropy for a given state or image embedding. Since APT does not itself perform representation learning, it requires an auxiliary representation learning loss to provide latent vectors for entropy estimation, although it is also possible to use random network embeddings [56]. We provide implementations of APT with the forward $g(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ and inverse dynamics $h(\mathbf{a}_t|\mathbf{z}_{t+1}, \mathbf{z}_t)$ representation learning losses:

$$r_t^{\text{APT}} \propto \sum_{j \in \text{random}} \log \|\mathbf{z}_t - \mathbf{z}_j\| \quad j = 1, \dots, K.$$

ProtoRL [68]: ProtoRL devises a self-supervised pre-training scheme that allows to decouple representation learning and exploration to enable efficient downstream generalization to previously unseen tasks. For this, ProtoRL uses the contrastive clustering assignment loss from SWaV [12] and learns latent representations and a set of prototypes to form the basis of the latent space. The prototypes are then used for more accurate estimation of entropy of the state-visitation distribution via KNN particle-based estimator:

$$r_t^{\text{Proto}} \propto \sum_{j \in \text{prototypes}} \log \|\mathbf{z}_t - \mathbf{z}_j\| \quad j = 1, \dots, K.$$

A.3 Competence-based Baselines

Competence-based approaches learn skills \mathbf{w} that maximize the mutual information between encoded observations (or states) and skills $I(\mathbf{z}; \mathbf{w})$. The mutual information has two decompositions $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{w}) - H(\mathbf{w}|\mathbf{z}) = H(\mathbf{z}) - H(\mathbf{z}|\mathbf{w})$. We provide baselines for both decompositions.

SMM [40]: SMM minimizes $D_{KL}(p_\pi(\mathbf{z}) \parallel p^*(\mathbf{z}))$, which maximizes the state entropy, while minimizing the cross entropy from the state to the target state distribution. When using skills, $H(\mathbf{z})$ can be rewritten as $H(\mathbf{z}|\mathbf{w}) + I(\mathbf{z}; \mathbf{w})$. $H(\mathbf{z}|\mathbf{w})$ can be maximized by optimizing the reward $r = \log q_{\mathbf{w}}(\mathbf{z})$, which is estimated using a VAE [36] that models the density of \mathbf{z} while executing skill \mathbf{w} . Similar to other mutual information methods that decompose $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{w}) - H(\mathbf{w}|\mathbf{z})$, SMM learns a discriminator $d(\mathbf{w}|\mathbf{z})$ over a set of discrete skills with a uniform prior that maximizes $H(\mathbf{w})$:

$$r_t^{\text{SMM}} \triangleq \log p^*(\mathbf{z}) - \log q_{\mathbf{w}}(\mathbf{z}) - \log p(\mathbf{w}) + \log d(\mathbf{w}|\mathbf{z}).$$

DIAYN [21]: DIAYN and similar algorithms such as VIC [24] and VALOR [1] are perhaps the best competence-based exploration algorithms. These methods estimate the mutual information through the first decomposition $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{w}) - H(\mathbf{w}|\mathbf{z})$. $H(\mathbf{w})$ is kept maximal by drawing $\mathbf{w} \sim p(\mathbf{w})$ from a discrete uniform prior distribution and the density $-H(\mathbf{w}|\mathbf{z})$ is estimated with a discriminator $\log q(\mathbf{w}|\mathbf{z})$.

$$r_t^{\text{DIAYN}} \propto \log q(\mathbf{w}|\mathbf{z}) + \text{const.}$$

APS [43]: APS is a recent leading mutual information exploration method that uses the second decomposition $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{z}) - H(\mathbf{z}|\mathbf{w})$. $H(\mathbf{z})$ is estimated with a particle estimator as in APT [42] while $H(\mathbf{z}|\mathbf{w})$ is estimated with successor features as in VISR [30].³

$$r_t^{\text{APS}} \propto r_t^{\text{APT}}(\mathbf{z}) + \log q(\mathbf{z}|\mathbf{w})$$

B Hyper-parameters

In Table 2 we present a common set of hyper-parameters used in our experiments, while in table 3 we list individual hyper-parameters for each method.

Table 2: A common set of hyper-parameters used in our experiments.

Common hyper-parameter	Value
Replay buffer capacity	10^6
Action repeat	1 states-based and 2 for pixels-based
Seed frames	4000
n -step returns	3
Mini-batch size	1024 states-based and 256 for pixels-based
Seed frames	4000
Discount (γ)	0.99
Optimizer	Adam
Learning rate	10^{-4}
Agent update frequency	2
Critic target EMA rate (τ_Q)	0.01
Features dim.	1024 states-based and 50 for pixels-based
Hidden dim.	1024
Exploration stddev clip	0.3
Exploration stddev value	0.2
Number pre-training frames	up to 2×10^6
Number fine-tuning frames	1×10^5

³In this benchmark, the generalized policy improvement(GPI) [3] that is used in Atari games for APS and VISR is not implemented for continuous control experiments.

Table 3: Per algorithm sets of hyper-parameters used in our experiments.

ICM hyper-parameter		Value
Representation dim.		512
Reward transformation		$\log(r + 1.0)$
Forward net arch.	$(\mathcal{O} + \mathcal{A}) \rightarrow 1024 \rightarrow 1024 \rightarrow \mathcal{O} $	ReLU MLP
Inverse net arch.	$(2 \times \mathcal{O}) \rightarrow 1024 \rightarrow 1024 \rightarrow \mathcal{A} $	ReLU MLP
Disagreement hyper-parameter		Value
Ensemble size		5
Forward net arch:	$(\mathcal{O} + \mathcal{A}) \rightarrow 1024 \rightarrow 1024 \rightarrow \mathcal{O} $	ReLU MLP
RND hyper-parameter		Value
Representation dim.		512
Predictor & target net arch.	$ \mathcal{O} \rightarrow 1024 \rightarrow 1024 \rightarrow 512$	ReLU MLP
Normalized observation clipping		5
APT hyper-parameter		Value
Representation dim.		512
Reward transformation		$\log(r + 1.0)$
Forward net arch.	$(512 + \mathcal{A}) \rightarrow 1024 \rightarrow 512$	ReLU MLP
Inverse net arch.	$(2 \times 512) \rightarrow 1024 \rightarrow \mathcal{A} $	ReLU MLP
k in NN		12
Avg top k in NN		True
ProtoRL hyper-parameter		Value
Predictor dim.		128
Projector dim.		512
Number of prototypes		512
Softmax temperature		0.1
k in NN		3
Number of candidates per prototype		4
Encoder target EMA rate (τ_{enc})		0.05
SMM hyper-parameter		Value
Skill dim.		4
Skill discrim lr		10^{-3}
VAE lr		10^{-2}
DIAYN hyper-parameter		Value
Skill dim		16
Skill sampling frequency (steps)		50
Discriminator net arch.	$512 \rightarrow 1024 \rightarrow 1024 \rightarrow 16$	ReLU MLP
APS hyper-parameter		Value
Representation dim.		512
Reward transformation		$\log(r + 1.0)$
Successor feature dim.		10
Successor feature net arch.	$ \mathcal{O} \rightarrow 1024 \rightarrow 1024 \rightarrow 10$	ReLU MLP
k in NN		12
Avg top k in NN		True
Least square batch size		4096

C Per-domain Individual Results

Individual fine-tuning results for each methods are shown in Figure 5. Furthermore, Figures 6 and 7 demonstrate individual results of states and pixels based fine-tuning performance as a function of pre-training steps for each considered method and task.

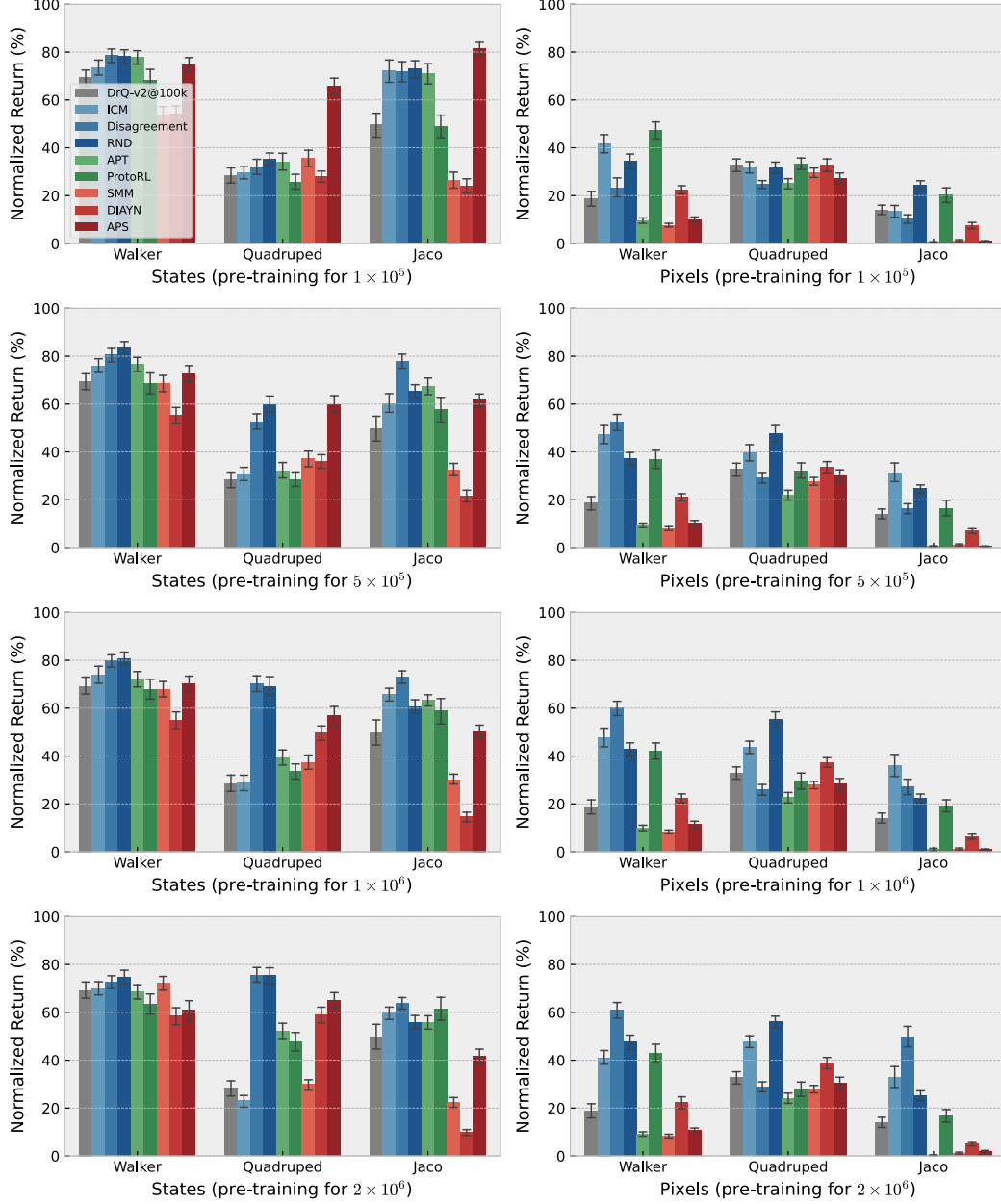


Figure 5: Individual results of fine-tuning for 100k steps after different degrees of pre-training for each considered method. The performance is aggregated across all the tasks within a domain and normalized with respect to the optimal performance.

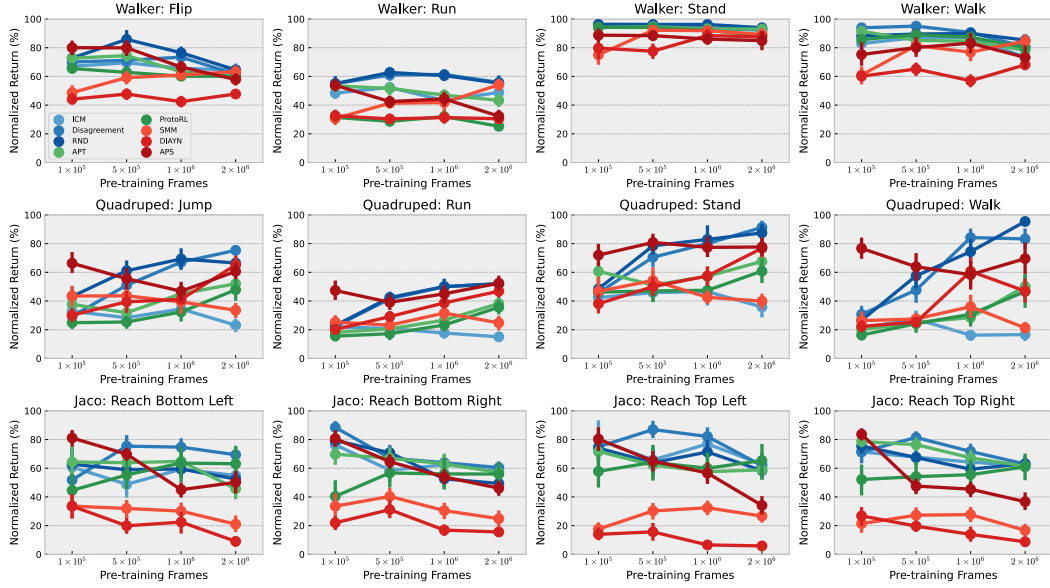


Figure 6: Individual results of fine-tuning efficiency as a function of pre-training steps for states-based learning.

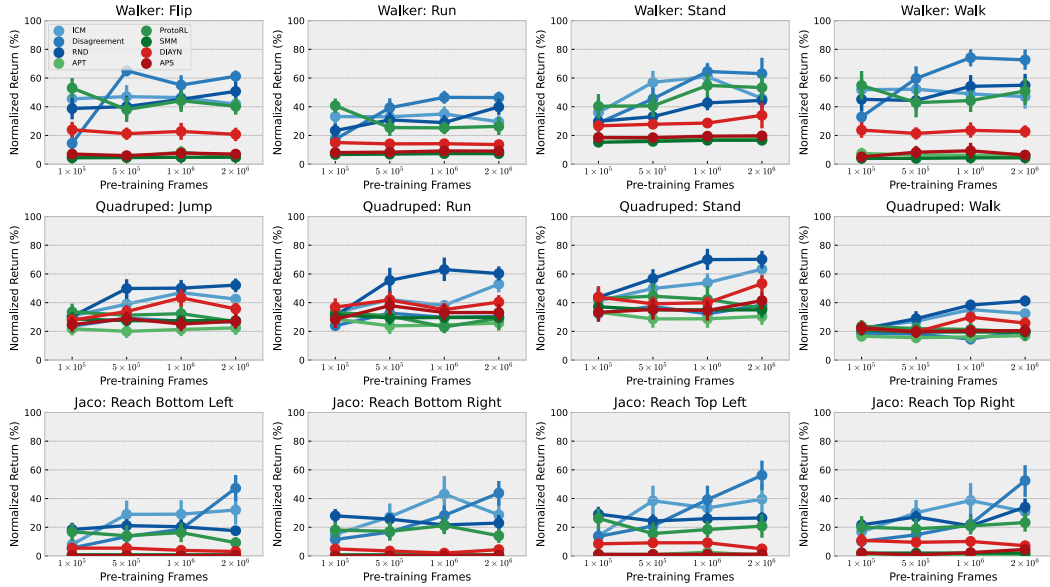


Figure 7: Individual results of fine-tuning efficiency as a function of pre-training steps for pixels-based learning.

D Finetuning Learning Curves

We provide finetuning learning curves for agents pre-trained for 2M steps with intrinsic rewards.

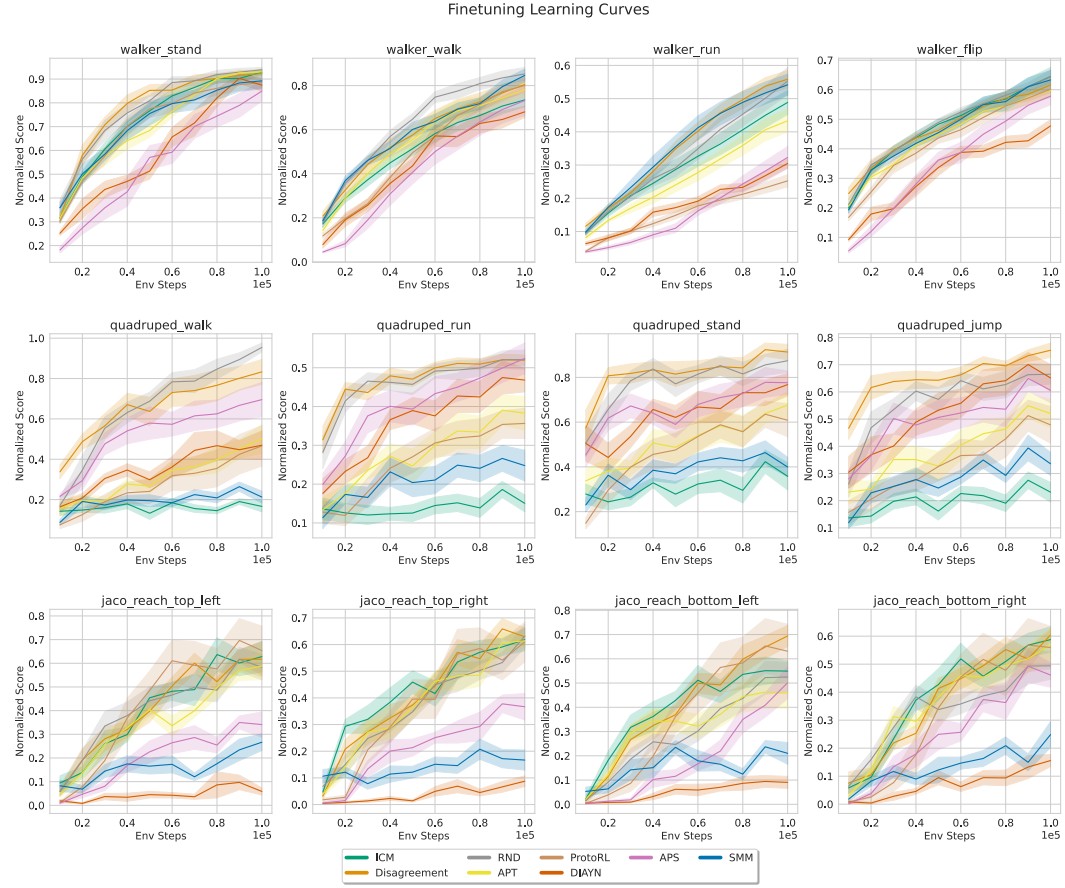


Figure 8: Finetuning curves for each evaluated unsupervised algorithm for each task considered in this benchmark after the agent has been pre-trained with intrinsic rewards.

E Individual Numerical Results

The individual numerical results of fine-tuning for each task and each method are presented in Table 4 for states-based learning, and in Table 5 for pixels-based learning.

Pre-training for 1×10^5 frames										
Domain	Task	DDPG (DrQ-v2)	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	535±19	559±20	581±21	580±28	523±16	388±36	352±22	638±33
	Run	325±25	384±24	437±24	437±33	424±28	250±34	244±28	259±26	428±26
	Stand	899±23	944±5	937±6	947±6	925±9	926±24	738±61	784±68	872±34
	Walk	748±47	805±46	911±14	857±32	888±19	831±31	592±53	584±25	731±70
Quadruped	Jump	236±48	291±35	261±45	383±57	334±48	220±33	386±56	267±34	589±57
	Run	157±31	195±31	198±40	203±20	161±27	138±21	224±33	179±26	420±49
	Stand	392±73	390±59	420±76	446±17	559±57	425±82	430±86	350±55	662±64
	Walk	229±57	185±20	265±45	229±26	173±29	141±23	227±47	193±29	664±56
Jaco	Reach bottom left	72±22	117±24	100±20	121±19	124±19	86±20	64±13	64±15	156±9
	Reach bottom right	117±18	155±10	179±6	161±8	141±14	82±21	68±17	44±8	164±10
	Reach top left	116±22	152±24	143±14	141±15	136±23	110±19	33±6	26±6	153±13
	Reach top right	94±18	159±15	159±15	168±9	175±6	116±22	47±12	59±12	186±7
Pre-training for 5×10^5 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	554±27	568±42	685±46	594±29	501±19	472±30	380±24	637±36
	Run	325±25	416±18	485±25	499±21	410±27	228±18	328±19	241±19	337±25
	Stand	899±23	930±7	940±8	946±5	930±5	925±17	906±18	762±44	869±27
	Walk	748±47	846±30	923±5	869±23	826±40	865±16	791±43	632±34	778±58
Quadruped	Jump	236±48	252±41	452±45	542±51	282±48	225±32	387±56	350±59	493±55
	Run	157±31	184±42	368±28	377±28	182±22	153±29	205±26	258±24	347±39
	Stand	392±73	422±49	649±53	722±49	470±80	433±63	499±78	459±54	743±46
	Walk	229±57	237±43	412±67	498±68	217±29	209±47	238±27	218±24	553±74
Jaco	Reach bottom left	72±22	94±16	145±13	113±11	123±15	106±18	61±10	38±9	134±6
	Reach bottom right	117±18	119±15	136±15	144±6	136±13	115±19	82±10	63±11	131±8
	Reach top left	116±22	125±18	165±9	121±16	118±18	122±23	57±9	29±10	124±11
	Reach top right	94±18	151±8	181±7	150±8	170±8	120±22	60±10	43±6	106±10
Pre-training for 1×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	524±20	586±36	610±27	505±22	480±17	486±17	338±23	531±24
	Run	325±25	344±30	488±23	482±23	373±22	254±29	332±40	249±24	352±31
	Stand	899±23	922±15	919±11	946±5	916±20	905±25	903±18	870±34	846±28
	Walk	748±47	845±27	880±19	873±24	821±35	848±29	746±51	553±33	808±61
Quadruped	Jump	236±48	306±42	595±39	615±53	400±53	287±52	349±63	365±15	415±46
	Run	157±31	157±24	444±39	444±42	237±35	206±32	280±40	343±25	400±48
	Stand	392±73	428±79	736±51	763±79	526±58	436±62	391±36	529±52	712±57
	Walk	229±57	140±24	729±46	644±70	246±33	266±64	312±63	525±76	505±84
Jaco	Reach bottom left	72±22	114±9	144±9	114±10	125±10	122±22	58±8	43±13	87±8
	Reach bottom right	117±18	126±10	129±10	106±13	128±12	113±20	62±9	34±6	109±9
	Reach top left	116±22	146±11	156±10	136±13	110±5	114±19	61±7	12±2	108±13
	Reach top right	94±18	143±10	159±10	132±10	149±11	123±21	61±9	31±9	101±9
Pre-training for 2×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	514±25	491±21	515±17	477±16	480±23	505±26	381±17	461±24
	Run	325±25	388±30	444±21	439±34	344±28	200±15	430±26	242±11	257±27
	Stand	899±23	913±12	907±15	923±9	914±8	870±23	877±34	860±26	835±54
	Walk	748±47	713±31	782±33	828±29	759±35	777±33	821±36	661±26	711±68
Quadruped	Jump	236±48	205±33	668±24	590±33	462±48	425±63	298±39	578±46	538±42
	Run	157±31	133±20	461±12	462±23	339±40	316±36	220±37	415±28	465±37
	Stand	392±73	329±58	840±33	804±50	622±57	560±71	367±42	706±48	714±50
	Walk	229±57	143±31	721±56	826±19	434±64	403±91	184±26	406±64	602±86
Jaco	Reach bottom left	72±22	106±8	134±8	101±12	88±12	121±22	40±9	17±5	96±13
	Reach bottom right	117±18	119±9	122±4	100±10	115±12	113±16	50±9	31±4	93±9
	Reach top left	116±22	119±12	117±14	111±10	112±11	124±20	50±7	11±3	65±10
	Reach top right	94±18	137±9	140±7	140±10	136±5	135±19	37±8	19±4	81±11

Table 4: Individual results of fine-tuning for 1×10^5 frames after different levels of pre-training in the states-based settings.

Pre-training for 1×10^5 frames										
Domain	Task	DrQ-v2	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	252±54	80±33	214±38	25±1	293±33	24±1	132±24	38±7
	Run	41±11	110±21	57±14	78±13	25±1	135±13	22±1	50±6	26±1
	Stand	212±28	315±67	250±62	261±34	162±9	353±67	133±8	233±22	162±9
	Walk	141±53	302±45	192±68	263±43	43±16	320±52	23±1	138±25	29±2
Quadruped	Jump	278±35	226±40	173±15	223±30	160±24	246±33	211±25	204±24	182±32
	Run	156±21	156±13	112±12	145±17	134±21	156±27	148±18	173±23	133±24
	Stand	309±47	329±49	259±31	350±43	266±37	342±35	297±36	350±48	265±48
	Walk	151±31	160±10	134±24	154±16	119±17	168±24	149±18	157±23	161±27
Jaco	Reach bottom left	23±10	18±7	12±4	41±7	0±0	38±11	1±1	12±4	0±0
	Reach bottom right	23±8	30±12	23±8	57±8	0±0	37±9	1±0	10±3	0±0
	Reach top left	40±9	31±11	30±9	66±9	0±0	59±14	2±1	19±4	2±1
	Reach top right	37±9	37±13	22±8	48±7	3±2	45±16	4±3	24±8	4±2
Pre-training for 5×10^5 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	260±39	360±16	222±37	28±2	210±44	25±1	117±18	32±2
	Run	41±11	110±15	131±19	103±13	26±1	85±16	23±1	47±4	27±1
	Stand	212±28	499±62	398±65	289±26	155±11	355±61	139±9	243±16	161±9
	Walk	141±53	305±51	348±46	258±33	37±10	250±54	23±1	125±19	48±19
Quadruped	Jump	278±35	286±50	214±24	366±44	147±26	229±42	201±20	248±28	212±27
	Run	156±21	198±29	153±19	261±38	112±20	144±27	138±16	197±24	178±23
	Stand	309±47	398±69	298±35	453±47	229±42	355±67	279±26	313±31	281±51
	Walk	151±31	193±31	129±20	206±30	111±20	157±25	139±13	140±19	141±24
Jaco	Reach bottom left	23±10	65±20	30±8	47±8	0±0	31±14	1±1	12±3	0±0
	Reach bottom right	23±8	56±16	34±10	52±6	0±0	35±11	1±0	7±2	0±0
	Reach top left	40±9	87±22	47±11	55±8	1±1	35±15	2±1	20±4	2±1
	Reach top right	37±9	68±17	33±5	61±8	2±1	42±14	4±3	21±5	1±1
Pre-training for 1×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	256±49	305±34	250±29	46±17	244±37	26±1	126±26	42±12
	Run	41±11	116±16	155±12	96±13	26±1	84±11	24±1	47±4	30±3
	Stand	212±28	534±73	565±37	374±37	150±13	480±63	145±6	251±19	170±10
	Walk	141±53	285±45	433±29	316±41	36±9	258±39	25±1	137±25	54±25
Quadruped	Jump	278±35	345±47	199±31	368±38	156±27	237±50	201±20	319±38	184±29
	Run	156±21	179±11	140±24	297±36	115±21	108±16	139±13	165±17	155±22
	Stand	309±47	430±44	257±35	559±51	229±42	338±71	279±26	319±27	275±48
	Walk	151±31	251±25	104±19	274±19	115±21	152±28	139±13	213±20	146±23
Jaco	Reach bottom left	23±10	65±19	42±10	46±9	0±0	36±13	1±1	8±3	0±0
	Reach bottom right	23±8	88±23	58±11	44±9	0±0	43±10	1±0	4±1	0±0
	Reach top left	40±9	76±19	89±19	59±7	6±4	41±10	2±1	20±4	2±0
	Reach top right	37±9	87±24	49±12	47±7	2±1	47±12	4±3	22±6	5±1
Pre-training for 2×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	231±34	339±16	280±31	28±2	223±27	26±1	114±21	38±9
	Run	41±11	98±11	154±9	133±15	25±2	87±18	24±1	45±3	30±3
	Stand	212±28	401±40	552±92	389±49	155±11	467±69	145±6	298±71	172±10
	Walk	141±53	274±44	424±36	321±43	35±8	297±48	25±1	132±19	37±5
Quadruped	Jump	278±35	312±18	194±21	383±28	164±27	197±35	201±20	262±22	199±29
	Run	156±21	249±21	143±25	284±18	121±20	137±35	139±13	190±18	156±24
	Stand	309±47	506±40	305±35	561±43	243±41	290±56	279±26	426±40	331±43
	Walk	151±31	231±15	145±10	294±20	122±21	138±35	139±13	184±23	146±24
Jaco	Reach bottom left	23±10	72±20	106±18	39±3	0±0	21±5	1±1	7±2	1±0
	Reach bottom right	23±8	58±19	90±15	47±9	0±0	28±7	1±0	9±3	1±1
	Reach top left	40±9	89±22	127±21	60±6	0±0	47±16	2±1	11±2	2±1
	Reach top right	37±9	69±18	118±23	76±11	1±1	52±12	4±3	16±3	10±3

Table 5: Individual results of fine-tuning for 1×10^5 frames after different levels of pre-training in the pixels-based settings.

F Compute Resources

URLB is designed to be accessible to the RL research community. Both state and pixel-based algorithms are implemented such that each algorithm requires a single GPU. For local debugging experiments we used NVIDIA RTX GPUs. For large-scale runs used to generate all results in this manuscripts, we used NVIDIA Tesla V100 GPU instances. All experiments were run on internal clusters. Each algorithm trains in roughly 30 mins - 12 hours depending on the snapshot (100k, 500k, 1M, 2M) and input (states, pixels). Since this benchmark required roughly 8k experiments (2 states / pixels, 12 tasks, 8 algorithms, 10 seeds, 4 snapshots) a total of 100 V100 GPUs were used to produce the results in this benchmark. Researchers who wish to build on URLB will, of course, not need to run this many experiments since they can utilize the results presented in this benchmark.

G Intuition on Competence-based Approaches Underperform on URLB

Across the three methods - data-based, knowledge-based, and competence-based - the best data-based and knowledge-based methods are competitive with one another. For instance, RND (a leading knowledge-based methods) and ProtoRL (a leading data-based method) achieve similar finetuning scores. Both are maximizing data diversity in two different ways - one through maximizing prediction error and the other through entropy maximization.

On the other hand, competence-based methods as a whole do much worse than data-based and knowledge-based ones. We hypothesize that this is due to current competence-based methods only supporting small skill spaces. Competence-based methods maximize a variational lower bound to the mutual information of the form:

$$I(\tau; z) = H(z) - H(z|\tau) = H(z) + \mathbb{E}[\log p(z|\tau)] \geq H(z) + \mathbb{E}[\log q(z|\tau)]$$

where $q(z|s)$ is called the discriminator. The discriminator can be interpreted as a classifier from $s \rightarrow z$ (or vice versa depending on how you decompose $I(s; z)$). In order to have an accurate discriminator, z is chosen to be small in practice (DIAYN - z is a 16 dim one-hot, SMM - z is 4 dim continuous, APS - z is 10 dim continuous).

OpenAI gym environments for continuous control mask this limitation because they terminate if the agent falls over and hence leak extrinsic signal about the downstream task into the environment. This means that the agent learns only useful behaviors that keep it balanced and therefore a small skill vector is sufficient for classifying these behaviors. However, in DeepMind control (and hence URLB) the episodes have fixed length and therefore the set of possible behaviors is much larger. If the skill space is too small, the most likely skills to be classified are different configurations of the agent lying on the ground. We hypothesize that building more powerful discriminators would improve competence-based exploration.