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 [47]: 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 [48]: 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:

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

RND [9]: 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 [41]: Active Pre-training (APT) utilizes a particle-based estimator [59] 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 [55]. 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:

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

ProtoRL [67]: 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 [11] 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:

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

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 [39]: 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 maximized by optimizing the reward $r = \log q_{\mathbf{w}}(\mathbf{z})$, which is estimated using a VAE [35] 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 [20]: DIAYN and similar algorithms such as VIC [23] 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 [42]: 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 [41] while $H(\mathbf{z}|\mathbf{w})$ is estimated with successor features as in VISR [29].[2]

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

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-turning frames	1×10^5

[2] In 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.	$(\lvert\mathcal{O}\rvert + \lvert\mathcal{A}\rvert) \to 1024 \to 1024 \to \lvert\mathcal{O}\rvert$ ReLU MLP
Inverse net arch.	$(2 \times \lvert\mathcal{O}\rvert) \to 1024 \to 1024 \to \lvert\mathcal{A}\rvert$ ReLU MLP
Disagreement hyper-parameter	**Value**
Ensemble size	5
Forward net arch:	$(\lvert\mathcal{O}\rvert + \lvert\mathcal{A}\rvert) \to 1024 \to 1024 \to \lvert\mathcal{O}\rvert$ ReLU MLP
RND hyper-parameter	**Value**
Representation dim.	512
Predictor & target net arch.	$\lvert\mathcal{O}\rvert \to 1024 \to 1024 \to 512$ ReLU MLP
Normalized observation clipping	5
APT hyper-parameter	**Value**
Representation dim.	512
Reward transformation	$\log(r + 1.0)$
Forward net arch.	$(512 + \lvert\mathcal{A}\rvert) \to 1024 \to 512$ ReLU MLP
Inverse net arch.	$(2 \times 512) \to 1024 \to \lvert\mathcal{A}\rvert$ 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
Intrinsic reward coefficient (α)	0.2
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.	$512 \to 1024 \to 1024 \to 16$ ReLU MLP
APS hyper-parameter	**Value**
Representation dim.	512
Reward transformation	$\log(r + 1.0)$
Successor feature dim.	10
Successor feature net arch.	$\lvert\mathcal{O}\rvert \to 1024 \to 1024 \to 10$ 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.

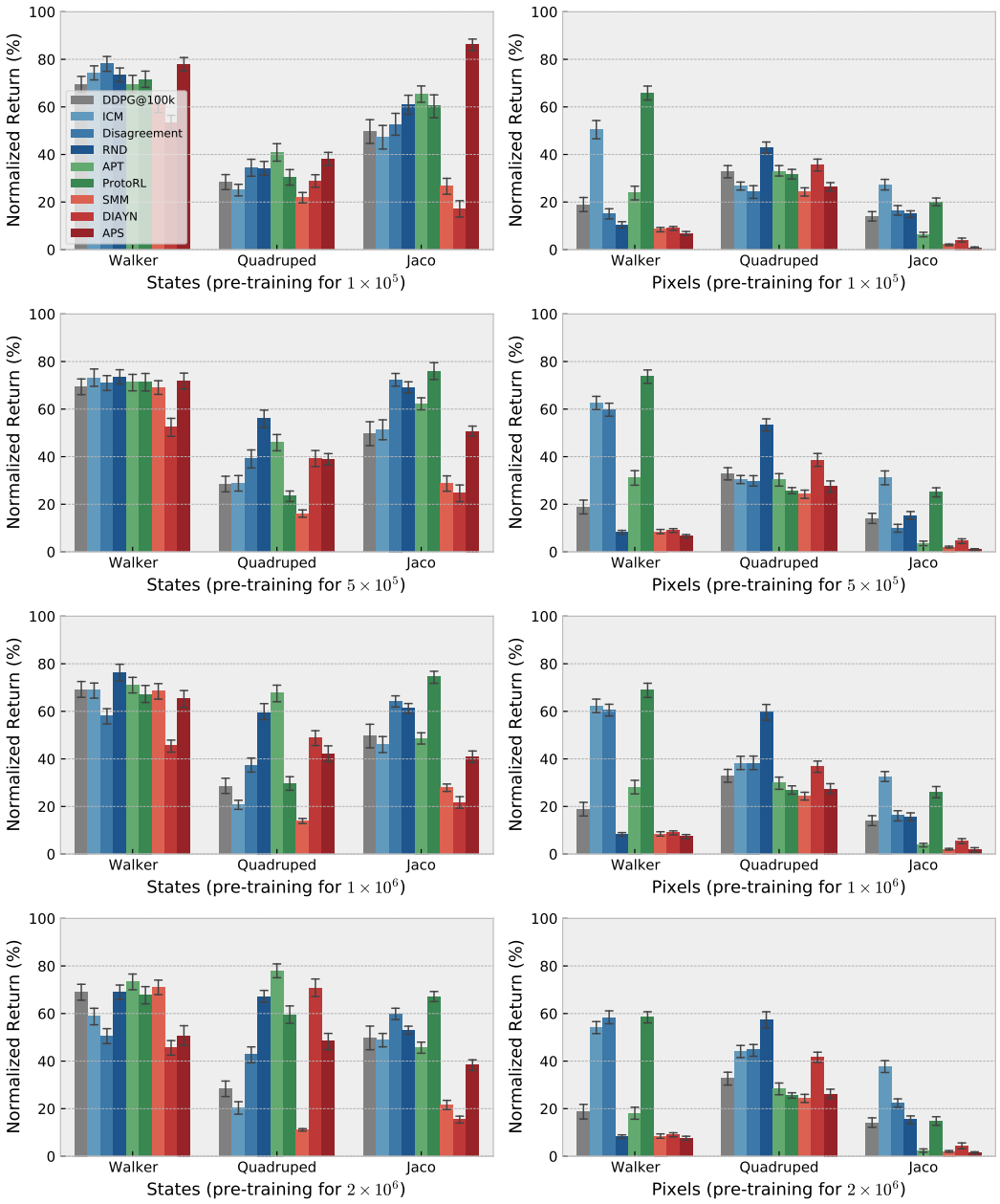

Figure 5: Individual results of fine-tuning for 100k steps after different degrees of pre-training for each considered method. The performance is aggregated across all the tasks within a domain and normalized with respect to the optimal performance.

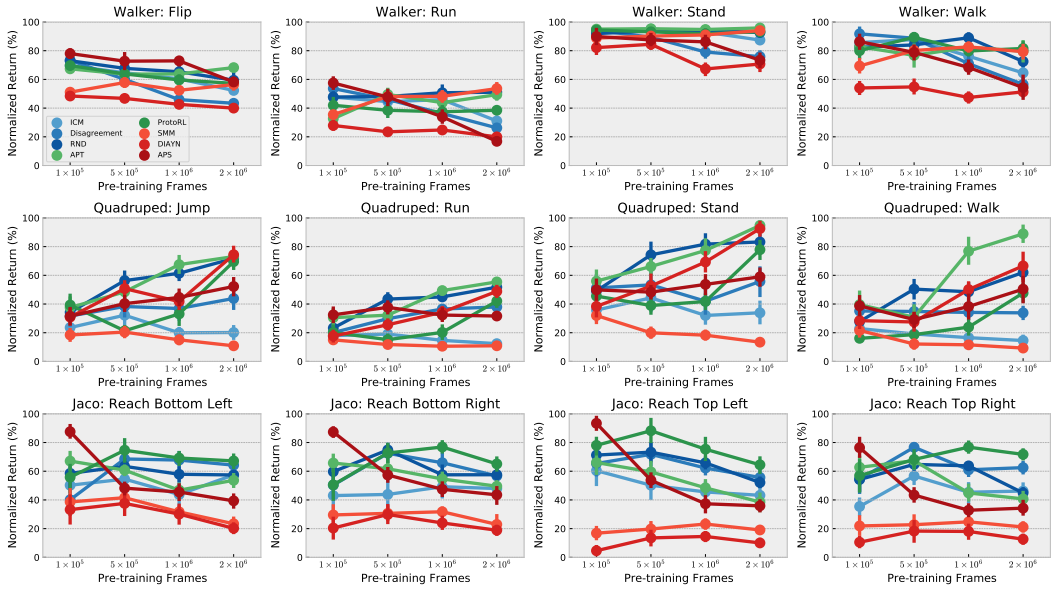

Figure 6: Individual results of fine-tuning efficiency as a function of pre-training steps for states-based learning.

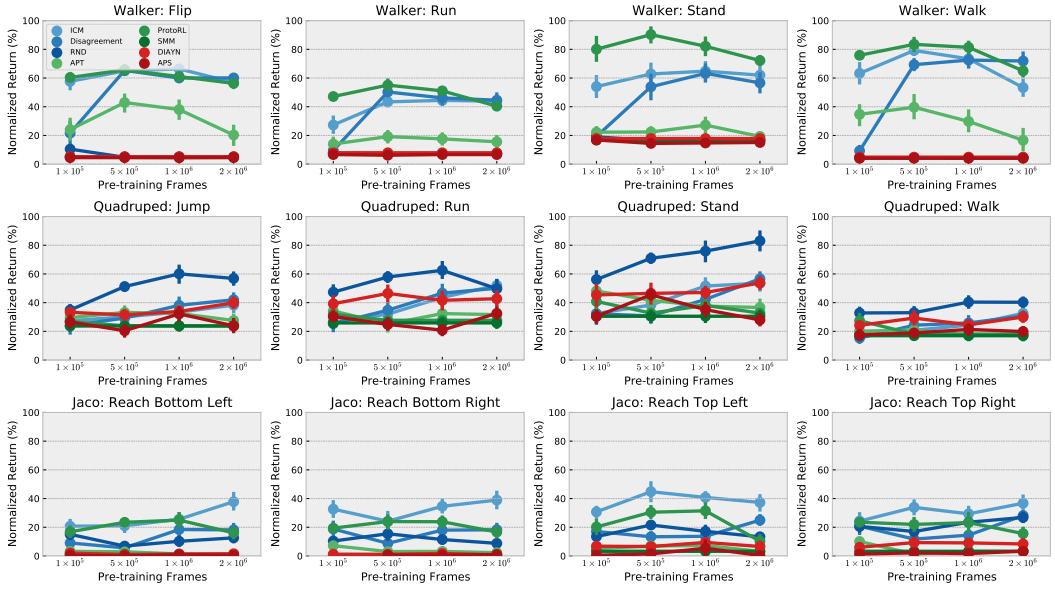

Figure 7: Individual results of fine-tuning efficiency as a function of pre-training steps for pixels-based learning.

D 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-trainining for 1×10^5 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	555±20	585±26	583±22	537±13	556±18	408±12	386±19	622±20
	Run	325±25	379±26	426±20	379±23	259±17	333±38	282±18	222±23	457±28
	Stand	899±23	926±9	926±9	904±13	934±9	928±17	873±38	808±43	884±47
	Walk	748±47	832±31	891±37	800±40	803±31	777±46	673±42	524±37	838±46
Quadruped	Jump	236±48	209±32	281±58	304±44	333±58	348±62	163±33	274±43	280±49
	Run	157±31	159±31	178±41	206±32	271±46	179±42	133±19	157±32	288±42
	Stand	392±73	329±65	471±78	452±60	515±69	419±62	294±44	352±60	458±61
	Walk	229±57	198±30	304±55	240±41	342±81	139±16	187±36	246±40	335±55
Jaco	Reach bottom left	72±22	97±22	77±20	113±19	129±12	107±18	74±16	64±18	169±8
	Reach bottom right	117±18	87±21	102±19	121±15	133±11	102±21	60±13	41±14	177±6
	Reach top left	116±22	115±19	124±16	136±10	126±16	149±11	32±7	8±5	178±7
	Reach top right	94±18	79±13	121±20	120±15	139±16	126±24	48±13	23±7	170±14
Pre-trainining for 5×10^5 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	516±21	479±19	540±27	508±21	509±18	461±24	373±6	580±43
	Run	325±25	354±34	366±28	382±20	393±29	306±36	382±24	186±9	379±40
	Stand	899±23	934±6	878±25	921±6	937±10	916±17	887±25	830±28	860±36
	Walk	748±47	852±35	861±29	816±34	745±73	867±20	778±25	532±43	765±44
Quadruped	Jump	236±48	285±56	339±57	500±59	430±38	189±27	181±29	450±44	357±44
	Run	157±31	169±39	264±44	385±35	286±26	134±27	104±15	226±28	334±35
	Stand	392±73	407±85	490±101	683±76	608±46	358±52	183±31	483±71	444±36
	Walk	229±57	165±31	301±53	436±56	266±71	161±23	105±25	238±51	252±41
Jaco	Reach bottom left	72±22	105±20	132±12	122±9	117±5	144±15	80±14	72±15	93±7
	Reach bottom right	117±18	89±19	147±12	152±7	125±11	147±12	62±9	60±12	116±9
	Reach top left	116±22	95±18	137±13	139±11	113±12	168±14	37±8	25±8	102±8
	Reach top right	94±18	126±11	171±5	144±7	150±11	152±13	50±13	40±15	96±9
Pre-trainining for 1×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	482±23	366±18	521±35	509±16	476±21	418±16	340±17	582±20
	Run	325±25	365±29	288±24	403±39	348±29	298±30	383±27	197±10	269±32
	Stand	899±23	914±10	777±34	926±8	932±6	902±27	894±13	661±36	847±39
	Walk	748±47	739±40	689±49	864±26	800±25	775±26	802±28	460±30	663±43
Quadruped	Jump	236±48	177±29	327±69	545±39	599±48	294±66	133±23	370±49	395±46
	Run	157±31	130±22	324±22	399±21	438±20	179±40	94±13	304±31	287±36
	Stand	392±73	294±50	386±85	752±63	711±72	384±44	168±25	637±60	494±59
	Walk	229±57	143±24	295±31	420±52	665±76	207±41	100±18	433±40	331±70
Jaco	Reach bottom left	72±22	84±17	130±10	111±9	90±8	133±8	61±6	58±12	87±10
	Reach bottom right	117±18	100±14	134±12	116±12	111±8	156±7	64±5	48±7	96±9
	Reach top left	116±22	87±12	118±8	125±6	92±9	143±14	44±5	27±5	71±10
	Reach top right	94±18	101±14	136±8	142±5	100±11	171±8	55±7	40±10	73±9
Pre-trainining for 2×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	417±16	346±13	474±39	544±14	456±12	450±24	319±17	465±20
	Run	325±25	247±21	208±15	406±30	392±26	306±13	426±26	158±8	134±16
	Stand	899±23	859±23	746±34	911±5	942±6	917±27	924±12	695±46	721±44
	Walk	748±47	627±42	549±37	704±30	773±70	792±41	770±44	498±27	527±79
Quadruped	Jump	236±48	178±35	389±62	637±12	648±18	617±44	96±7	660±43	463±51
	Run	157±31	110±18	337±30	459±6	492±14	373±33	96±6	433±29	281±17
	Stand	392±73	312±68	512±89	766±43	872±23	716±56	123±11	851±43	542±53
	Walk	229±57	126±27	293±37	536±39	770±47	412±54	80±6	576±81	436±79
Jaco	Reach bottom left	72±22	111±11	124±7	110±5	103±8	129±8	45±7	39±6	76±8
	Reach bottom right	117±18	97±9	115±10	117±7	100±6	132±8	46±11	38±5	88±11
	Reach top left	116±22	82±14	106±12	99±6	73±12	123±9	36±3	19±4	68±6
	Reach top right	94±18	103±11	139±7	100±6	90±10	159±7	47±6	28±6	76±10

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

E 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

Pre-trainining for 1×10^5 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	319±29	119±36	57±15	133±40	334±8	28±3	28±0	25±2
	Run	41±11	91±19	31±6	25±2	46±12	157±6	24±2	26±0	22±2
	Stand	212±28	473±63	169±25	166±22	195±28	701±75	147±16	156±1	147±7
	Walk	141±53	369±43	54±19	26±2	202±39	442±11	26±2	28±0	23±2
Quadruped	Jump	278±35	200±24	176±37	256±23	217±23	196±21	174±21	244±18	195±28
	Run	156±21	125±14	119±24	223±20	161±19	150±16	121±15	184±22	143±14
	Stand	309±47	258±28	257±57	448±42	384±25	326±40	243±30	362±57	243±32
	Walk	151±31	142±16	109±21	235±25	142±17	196±29	121±15	173±11	125±23
Jaco	Reach bottom left	23±10	46±8	20±7	34±6	7±2	38±7	2±1	4±1	1±0
	Reach bottom right	23±8	67±10	38±10	21±3	14±4	39±5	1±0	2±1	1±0
	Reach top left	40±9	69±8	38±6	30±5	11±3	45±10	7±1	15±5	1±1
	Reach top right	37±9	54±12	46±11	46±7	22±5	53±4	7±1	13±5	2±1
Pre-trainining for 5×10^5 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	359±9	361±12	26±0	237±31	363±16	28±3	28±0	25±2
	Run	41±11	144±9	167±7	24±0	64±12	183±10	24±2	26±0	20±1
	Stand	212±28	550±60	472±74	144±4	196±24	791±43	147±16	156±1	126±9
	Walk	141±53	462±16	404±20	26±0	231±47	487±24	26±2	28±0	23±2
Quadruped	Jump	278±35	221±28	214±38	376±19	244±28	175±13	174±21	231±23	150±28
	Run	156±21	150±13	163±19	272±14	118±23	130±9	121±15	218±25	117±17
	Stand	309±47	306±28	245±36	566±23	333±53	261±19	243±30	371±53	363±26
	Walk	151±31	148±12	173±17	236±26	149±26	129±9	121±15	209±33	135±15
Jaco	Reach bottom left	23±10	47±9	12±5	15±3	6±5	53±5	2±1	1±0	0±0
	Reach bottom right	23±8	49±12	18±7	31±7	6±2	49±8	0±0	2±0	1±0
	Reach top left	40±9	101±14	30±6	49±5	15±5	69±8	7±1	14±2	3±3
	Reach top right	37±9	76±10	26±9	38±6	2±1	49±9	7±1	21±6	4±2
Pre-trainining for 1×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	367±12	332±9	26±0	210±33	337±7	28±3	28±0	24±2
	Run	41±11	148±10	154±14	24±0	58±12	170±6	24±2	26±0	22±2
	Stand	212±28	567±50	554±48	144±4	238±44	719±53	147±16	156±1	129±16
	Walk	141±53	428±32	423±28	26±0	173±41	475±22	26±2	28±0	23±2
Quadruped	Jump	278±35	245±35	280±38	441±41	238±41	175±13	174±21	249±25	236±31
	Run	156±21	205±26	219±25	294±24	152±24	130±9	121±15	196±27	98±14
	Stand	309±47	412±38	337±46	606±54	300±38	304±41	243±30	376±29	280±37
	Walk	151±31	174±25	185±34	289±26	124±23	129±9	121±15	175±18	152±23
Jaco	Reach bottom left	23±10	57±8	41±9	23±7	3±2	56±10	2±1	2±1	1±1
	Reach bottom right	23±8	71±8	36±14	23±5	6±2	49±6	0±0	3±1	0±0
	Reach top left	40±9	92±7	31±7	39±8	16±3	71±11	7±1	21±5	12±11
	Reach top right	37±9	66±10	32±7	53±5	7±4	52±14	7±1	20±4	3±2
Pre-trainining for 2×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	315±20	332±9	26±0	112±36	311±5	28±3	28±0	25±2
	Run	41±11	146±11	148±14	24±0	52±13	134±7	24±2	26±0	22±2
	Stand	212±28	543±49	496±58	144±4	169±19	632±24	147±16	156±1	132±13
	Walk	141±53	311±33	419±34	26±0	97±39	379±19	26±2	28±0	23±2
Quadruped	Jump	278±35	281±34	308±30	418±29	202±29	175±13	174±21	293±24	175±31
	Run	156±21	245±16	235±21	234±27	149±27	130±9	121±15	201±21	153±17
	Stand	309±47	428±42	452±34	663±55	292±37	261±19	243±30	429±32	223±30
	Walk	151±31	233±23	219±17	288±22	122±23	129±9	121±15	214±23	142±19
Jaco	Reach bottom left	23±10	85±12	41±8	28±6	1±0	36±8	2±1	3±1	0±0
	Reach bottom right	23±8	80±10	37±7	18±4	4±1	34±7	0±0	1±0	2±0
	Reach top left	40±9	84±11	56±6	30±5	7±1	23±9	7±1	15±6	0±0
	Reach top right	37±9	82±12	63±6	60±6	7±5	35±8	7±1	18±7	7±2

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

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.