REFERENCES

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Bram Bakker. Reinforcement learning with long short-term memory. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pp. 1475–1482, Cambridge, MA, USA, 2001. MIT Press.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, Jun 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://dx.doi.org/10.1613/jair.3912.

Homanga Bharadhwaj, Shoichiro Yamaguchi, and Shin ichi Maeda. Manga: Method agnostic neural-policy generalization and adaptation, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pp. 2249–2257, 2011.

Maxime Chevalier-Boisvert. gym-miniworld environment for openai gym. https://github.com/maximecb/gym-miniworld, 2018. MiniWorld licensed under Apache License 2.0.

Samuel Choi, Dit-Yan Yeung, and Nevin Zhang. Hidden-mode markov decision processes for nonstationary sequential decision making. pp. 264–287, 01 2001.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl$ˆ2$: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016. URL http://arxiv.org/abs/1611.02779.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv e-prints*, art. arXiv:1703.03400, Mar 2017.

Aurélien Garivier, Tor Lattimore, and Emilie Kaufmann. On explore-then-commit strategies. In *NIPS*, pp. 784–792, 2016.

Abhishek Gupta, Russell Mendonca, Yuxuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *NeurIPS*, pp. 5307–5316, 2018.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin A. Riedmiller. Learning an embedding space for transferable robot skills. In *ICLR (Poster)*. OpenReview.net, 2018.

Nicolas Heess, Jonathan J. Hunt, Timothy P. Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *CoRR*, abs/1512.04455, 2015. URL http://arxiv.org/abs/1512.04455.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning, 2017.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A. Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv e-prints*, art. arXiv:1905.06424, May 2019.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pp. 143–173. Springer, 2012.

Evan Zheran Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Explore then execute: Adapting without rewards via factorized meta-reinforcement learning, 2020.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J. Pal, and Liam Paull. Active domain randomization, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1602.01783, Feb 2016.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2701–2710. PMLR, 2017.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *NIPS*, pp. 4026–4034, 2016.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. URL http://jmlr.org/papers/v20/18-339.html.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *arXiv e-prints*, art. arXiv:1710.06537, Oct 2017.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. *arXiv e-prints*, art. arXiv:1909.09157, Sep 2019.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340. PMLR, 2019.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Matthew E. Taylor and Peter Stone. An introduction to inter-task transfer for reinforcement learning. *AI Magazine*, 32(1):15–34, 2011. URL http://www.cs.utexas.edu/users/ai-lab?AAAIMag11-Taylor.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *CoRR*, abs/1707.04175, 2017. URL http://arxiv.org/abs/1707.04175.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world, 2017.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 1015–1022, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273624. URL https://doi.org/10.1145/1273496.1273624.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification, 2017.

Wenhao Yu, C. Karen Liu, and Greg Turk. Policy transfer with strategy optimization, 2018.

Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies. 2019.

Luisa Zintgraf, Maximilian Igl, Kyriacos Shiarlis, Anuj Mahajan, Katja Hofmann, and Shimon Whiteson. Variational Task Embeddings for Fast Adaptation in Deep Reinforcement Learning. *arXiv e-prints*, 2019.

A THE IMPORT ALGORITHM

The algorithm is described in details in Algorithm 2. In our implementation, the value function network used for **(A)** and **(B)** is the same, i.e. shared. We specialize the input, i.e. for **(A)** the input will be $(s_t, f_H(\tau_t))$ and $(s_t, f_\mu(\mu_t))$ for **(B)**.

Algorithm 2 Details of IMPORT Training

Initialize $\sigma, \omega, \theta, \nu$ arbitrarily
Hyperparameters: Number of iterations K, Number of transitions per update steps M, discount factor γ, GAE parameter γ_{GAE}, Adam learning rate η, weighting of the **(C)** objective β, weighting of the entropy objective λ_h, weighting of the critic objective λ_c
$Optim = Adam(\eta)$
for $k = 1, \dots, K$ **do**
  if k is odd **then**
    Collect M transitions according to π_H in buffer B_H.
  else
    Collect M transitions according to π_μ in buffer B_μ.
  end if
  $\delta_\sigma, \delta_\omega, \delta_\theta = 0, 0, 0$

  $R^\mu \leftarrow \text{compute\_gae\_returns}(B_\mu, \gamma_{GAE})$
  $R^H \leftarrow \text{compute\_gae\_returns}(B_H, \gamma_{GAE})$

  $\delta_{\theta,\omega} \mathrel{+}= \frac{1}{|B_H|} \sum_{b \in B_H} \sum_{t=1}^{T} [R_t^{\mu,b} - V_\nu(s_t^b, z_t^b)] \nabla_{\theta,\omega} \log \pi_H(a_t^b | s_t^b, z_t^b)$
  $\delta_{\theta,\omega} \mathrel{+}= \frac{\lambda_h}{|B_H|} \sum_{b \in B_H} \sum_{t=1}^{T} \nabla_{\theta,\omega} H\big(\pi_H(a_t^b | s_t^b, z_t^b)\big)$
  $\delta_\omega \mathrel{-}= \frac{2\beta}{|B_H|} \sum_{b \in B_H} \sum_{t=1}^{T} [f_H^\omega(s_t^b, z_t^b) - f_\mu(s_t^b, \mu_t^b)] \nabla_\omega f_H^\omega(s_t^b, z_t^b)$
  $\delta_\nu \mathrel{-}= \frac{2\lambda_c}{|B_H|} \sum_{b \in B_H} \sum_{t=1}^{T} [R_t^{H,b} - V_\nu(s_t^b, z_t^b)] \nabla_\nu V_\nu(s_t^b, z_t^b)$

  $\delta_{\theta,\sigma} \mathrel{+}= \frac{1}{|B_\mu|} \sum_{b \in B_\mu} \sum_{t=1}^{T} [R_t^{H,b} - V_\nu(s_t^b, \mu_t^b)] \nabla_{\theta,\sigma} \log \pi_\mu(a_t^b | s_t^b, \mu_t^b)$
  $\delta_{\theta,\sigma} \mathrel{+}= \frac{\lambda_h}{|B_\mu|} \sum_{b \in B_\mu} \sum_{t=1}^{T} \nabla_{\theta,\sigma} H\big(\pi_\mu(a_t^b | s_t^b, \mu_t^b)\big)$
  $\delta_\nu \mathrel{-}= \frac{2\lambda_c}{|B_\mu|} \sum_{b \in B_\mu} \sum_{t=1}^{T} [R_t^{\mu,b} - V_\nu(s_t^b, \mu_t^b)] \nabla_\nu V_\nu(s_t^b, \mu_t^b)$

  $\theta \leftarrow Optim(\theta, \delta_\theta)$
  $\omega \leftarrow Optim(\omega, \delta_\omega)$
  $\sigma \leftarrow Optim(\sigma, \delta_\sigma)$
  $\nu \leftarrow Optim(\nu, \delta_\nu)$
end for

B IMPLEMENTATION DETAILS

B.1 DATA COLLECTION AND OPTIMIZATION

We focus on on-policy training for which we use the actor-critic method A2C (Mnih et al., 2016) algorithm with generalized advantage estimation. We use a distributed execution to accelerate experience collection. Several worker processes independently collect trajectories. As workers progress, a shared replay buffer is filled with trajectories and an optimization step happens when the buffer's capacity bs is reached. After model updates, replay buffer is emptied and the parameters of all workers are updated to guarantee synchronisation.

B.2 NETWORK ARCHITECTURES

The architecture of the different methods remains the same in all our experiments, except that the number of hidden units changes across considered environments and we consider convolutional neural networks for the Maze3d environment. A description of the architectures of each method is given in Fig. 2.
Unless otherwise specified, MLP blocks represent single linear layers activated with a $tanh$ function and their output size is hs. All methods aggregate the trajectory into an embedding z_t using a GRU with hidden size hs. Its input is the concatenation of representations of the last action a_{t-1} and

HPs	CartPole	Acrobot	Bandits	TMDP	Maze3d
E	16	128	16	16	32
Tr	20	20	20	20	20
hs	16	32	16	64	32
hs_μ	$\{2, 4, 8, 16\}$	$\{2, 4, 8, 16\}$	16	$\{16, 32\}$	$\{2, 16, 32\}$
γ	0.95	0.95	0.90	0.90	0.90
λ_h	$\{1., 1e^{-1}\}$				$\{1e^{-1}, 1e^{-2}, 1e^{-3}\}$
γ_{GAE}	$\{0.0, 1.0\}$				
clip gradient	40				
η	$\{1e^{-3}, 3e^{-4}\}$				
λ_c	$\{1., 1e^{-1}, 1e^{-2}\}$				
β	$\{1e^{-1}, 1e^{-2}, 0.\}$				

Table 4: Hyperparameters tested per environments. At each training epoch, we run our agent on E environments in parallel collecting Tr transitions on each of them resulting in batches of $M = E * Tr$ transitions.

current state s_t obtained separately. Actions are encoded as one-hot vectors. When episodes begin, we initialize the last action with a vector of zeros. For bandits environments, the current state corresponds to the previous reward. TS uses the same GRU architecture to aggregate the history into z_t.

All methods use a $softmax$ activation to obtain a probability distribution over actions.
The use of the hidden-state z_t differs across methods. While **RNNs** only use z_t as an input to the policy and critic, both **TS** and **TI** map z_t to a belief distribution that is problem-specific, e.g. Gaussian for control problems, Beta distribution for bandits, and a multinomial distribution for Maze and CartPole-task environments. For instance, z_t is mapped to a Gaussian distribution by using two MLPs whose outputs of size $|\mu|$ correspond to the mean and variance. The variance values are mapped to $[0, 1]$ using a $sigmoid$ activation.

IMPORT maps z_t to an embedding f_H, whereas the task embedding f_μ is obtained by using a $tanh$-activated linear mapping of μ_t. Both embeddings have size hs_μ, tuned by cross-validation onto a set of validation tasks. The input of the shared policy head ϕ is the embedding associated with the policy to use, i.e. either f_H when using π_H or f_μ when using f_μ.

For the Maze3d experiment and in all methods, we pre-process the pixel input s_t with three convolutional layers (with output channels 32, stride is 2 and respective kernel sizes are 5, 5 and 4) and LeakyReLU activation. We also use a batch-norm after each convolutional layer. The output is flattened, linearly mapped to a vector of size hs and $tanh$-activated.

C EXPERIMENTS

In this section, we explain in deeper details the environments and the set of hyper-parameters we considered. We add learning curves of all experiments to supplement results from Table 1, 2, 3 and 5 in order to study sample efficiency.

Task descriptor. Note that for CartPole and Acrobot μ is normalized to be in $[-1, 1]^D$ where D is the task descriptor dimension. The task distribution q is always uniform, see the description of the environments for details. For experiments with task identifiers, we associate to each sampled task an integer value corresponding to the order of generation, and encode it usong a one-hot vector.

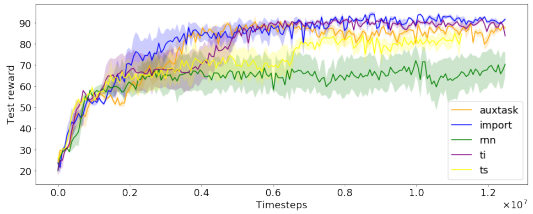
Hyperparameters. Hyperparameter ranges are specified in Table 4. For TS, we consider sampling μ from the posterior dynamics distribution every k steps with $k \in \{1, 5, 10, 20\}$.

C.1 CARTPOLE.

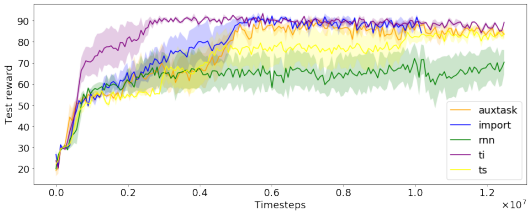
We consider the classic CartPole control environment where the environment dynamics change within a set $\mathcal{M}$ ($|\mu| = 5$) described by the following physical variables: gravity, cart mass, pole mass, pole length, magnetic force. Their respective pre-normalized domains are $[4.8, 14.8]$, $[0.5, 1.5]$, $[0.01, 0.19]$, $[0.2, 0.8]$, and $[-10, 10]$. The value of μ are uniformly sampled. Knowing some components of μ might not be required to behave optimally. The discrete action space is $\{-1, 1\}$.

Episode length is $T = 100$.

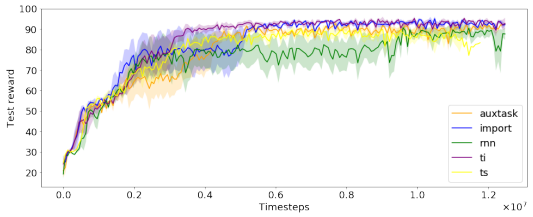
Final performance and sample efficiency. Table 1 shows IMPORT's performance is marginally superior to other methods in most settings. Learning curves in Figure 7 allow analyzing the sample efficiency of the different methods. Overall, IMPORT is more sample efficient than other methods in the privileged information μ setting. Moreover, the use of the auxiliary loss ($\beta > 0$) usually speed-up the learning convergence by enforcing the RNN to quickly produce a coherent embedding. We can see that only sharing parameters ($\beta = 0$) already helps improving over RNNs.



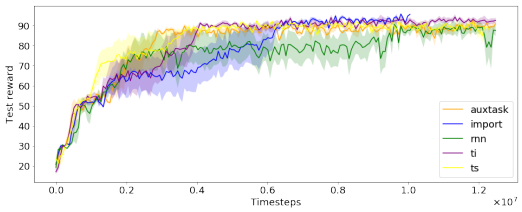
(a) CartPole with μ and $N = 10$



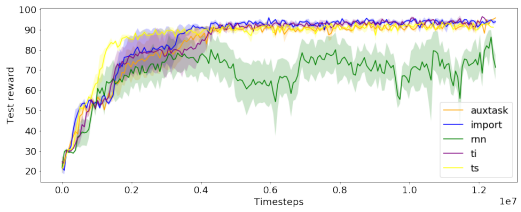
(b) CartPole with TID and $N = 10$



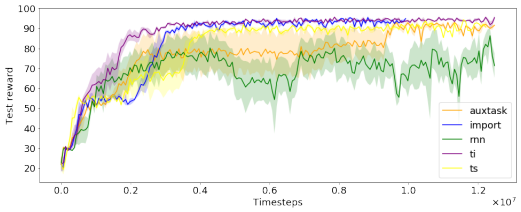
(c) CartPole with μ and $N = 20$



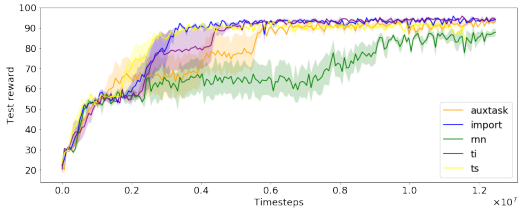
(d) CartPole with TID and $N = 20$



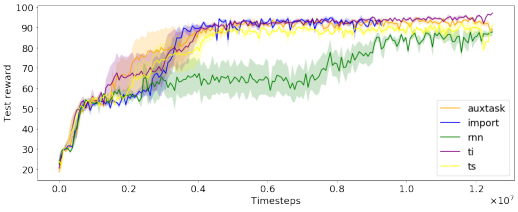
(e) CartPole with μ and $N = 50$



(f) CartPole with TID and $N = 50$



(g) CartPole with μ and $N = 100$



(h) CartPole with TID and $N = 100$

Figure 7: Evaluation on CartPole

Non-stationary environments. We consider the non-stationary version of CarPole environment where at each timestep, there is a probability $\rho = 0.05$ to sample a new dynamic μ. Table 8 shows that the performance of IMPORT, AuxTask and TI are comparable in these settings.

Method	$N = 10$	$N = 100$
AuxTask	86.4(1.0)	93.0(0.3)
IMPORT	**91.7(0.5)**	92.7(0.8)
RNN	65.5(4.3)	89.5(0.6)
TI	88.2(3.9)	**95.5(0.8)**
TS	86.7(1.6)	92.2(0.7)

Figure 8: CartPole (non-stationary).

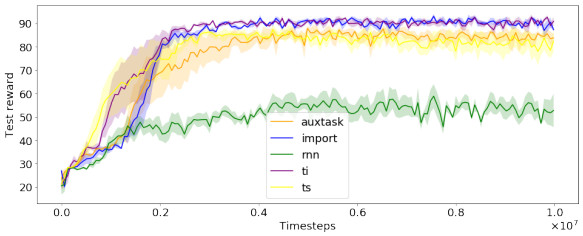


Figure 9: Non-stationary CartPole with $N = 10$

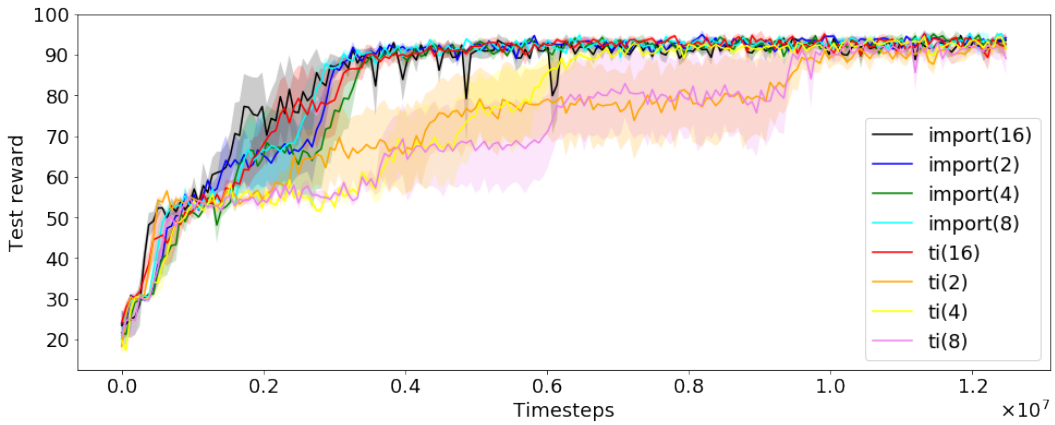
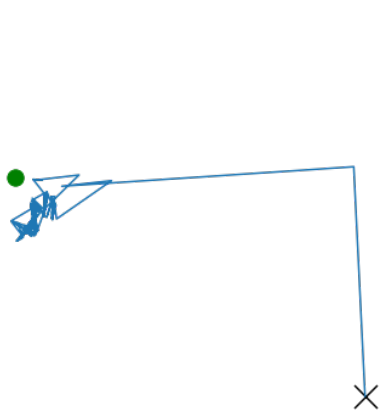


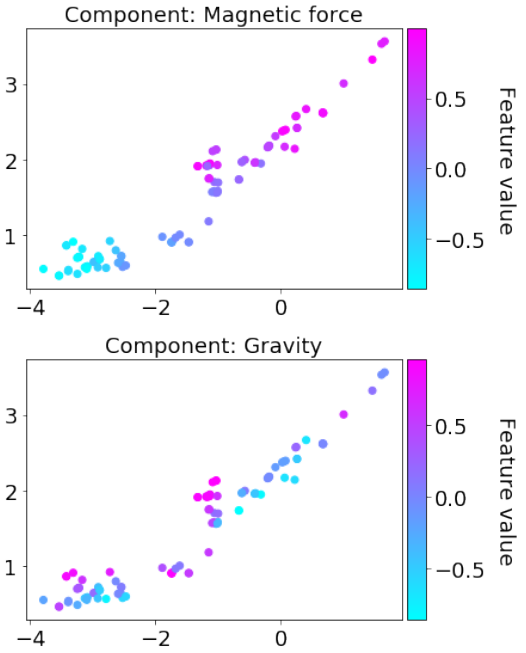
Figure 10: IMPORT and TI with different task embedding representation size on CartPole with $N = 20$

Size of built embeddings. We now study the impact of the task embedding representation size. As can be seen from Figure 10, IMPORT's performance remains stable for different representation sizes in $\{2, 4, 8, 16\}$ whereas TI's sample efficiency decreases with this dimension.

Trajectory and task embeddings. In Figure 11, we plot both the evolution of $f_H(\tau_t)$ during an episode of the final model obtained training IMPORT with two-dimensional task embeddings on CartPole with **task identifiers** (left) and task embedding $f_\mu(\mu)$ learnt by the informed policy (right). As expected, the history embedding gets close to the task embedding after just a few timesteps (left). Interestingly, task embeddings $f_\mu(\mu)$ are able to capture relevant information from the task. For instance, they are highly correlated with the *magnetic force* which is a very strong factor to "understand" from each new environment to control the system correctly. At the opposite, *gravity* is less correlated since it does not influence the optimal policy – whatever the gravity is, if the pole is on the left, then you have to go right and vice-versa.



(a) Value of $f_H(\tau_t)$ among episodes steps on CartPole with task identifiers. The green circle is the value of $f_\mu(\mu)$. The image shows that IMPORT starts with a random embedding, and is able to discover the task embedding with a reasonable performance in a few steps.



(b) Task embeddings $f_\mu(\mu)$ for Cartpole with task identifiers. The color of the point corresponds to the value of one of the 'real' physics component of the environment (unknown to the model).

Figure 11: Visualization of task embeddings upon Cartpole

C.2 ACROBOT

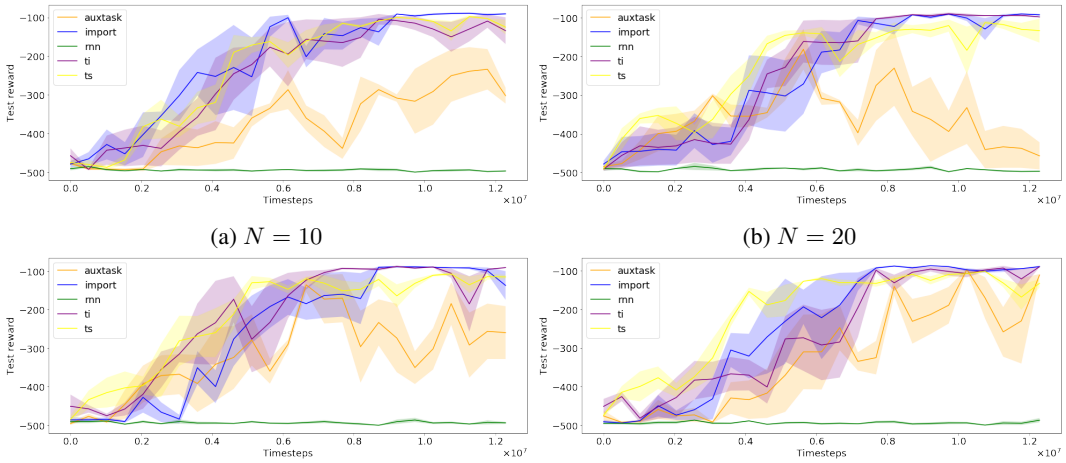
	$N = 10$	$N = 20$	$N = 50$	$N = 100$
AuxTask	$-189.0(54.8)$	$-98.3(1.8)$	$-103.0(8.0)$	$-93.6(1.3)$
IMPORT	$\mathbf{-87.2(0.9)}$	$\mathbf{-92.5(1.3)}$	$-88.9(1.1)$	$-88.9(1.6)$
RNN	$-483.6(1.6)$	$-482.7(4.0)$	$-480.7(3.5)$	$-485.0(3.7)$
TI	$-89.7(1.2)$	$-94.6(0.7)$	$\mathbf{-87.8(0.8)}$	$\mathbf{-87.3(1.2)}$
TS	$-101.4(2.0)$	$-102.1(6.0)$	$-102.4(2.0)$	$-102.3(0.8)$

Table 5: Acrobot

Acrobot consists of two joints and two links, where the joint between the two links is actuated. Initially, the links are hanging downwards, and the goal is to swing the end of the lower link up to a given height. Environment dynamics are determined by the length of the two links, their masses, their maximum velocity. Their respective pre-normalized domains are $[0.5, 1.5]$, $[0.5, 1.5]$, $[0.5, 1.5]$, $[0.5, 1.5]$, $[3\pi, 5\pi]$ and $[7\pi, 11\pi]$. Unlike CartPole, the environment is stochastic because the simulator applies noise to the applied force. The action space is $\{-1, 0, 1\}$. We also add an extra dynamics parameter which controls whether the action order is inverted, i.e. $\{1, 0, -1\}$, thus $|\mu| = 7$.

Episode length is 500.

IMPORT outperforms all baselines in settings with small training task sets (Figure 12 and Table 5) and perform similarly to TI on larger training task sets.



(a) $N = 10$ (b) $N = 20$

(c) $N = 50$ (d) $N = 100$

Figure 12: Performance on Acrobot

C.3 BANDITS

The **Bandit** environment is a standard Bernoulli multi-armed bandit problem with K arms. The vector $\mu \in \mathbb{R}^K$ denotes the probability of success of the independent Bernoulli distributions. Each dimension of μ is sampled uniformly between 0 and 0.5, the best arm is randomly selected and associated to a probability of 0.9. Although relatively simple, this environment assesses the ability of algorithms to learn nontrivial exploration/exploitation strategies.

Note that it is not surprising that UCB outperforms the other algorithms in this setting. UCB is an optimal algorithm for MAB and we have optimized it for achieving the best empirical performance. Moreover, IMPORT cannot leverage correlations between tasks since, due to the generation process, tasks are independent.

We visualize the task embeddings learnt by the informed policy in 13.

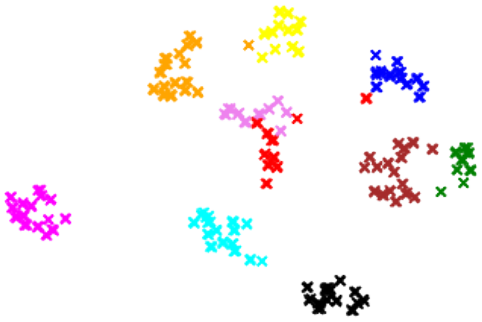
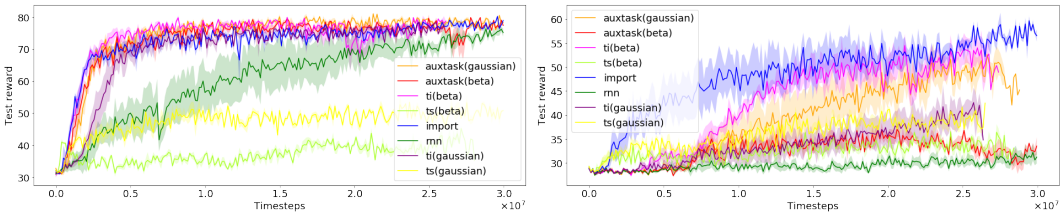


Figure 13: t-SNE of the task embeddings on the bandit problem with $K = 10$.



(a) Bandits with $K = 10$ arms

(b) Bandits with $K = 20$ arms

Figure 14: Learning curves on the bandit problem.

C.4 MAZE3D ENVIRONMENT

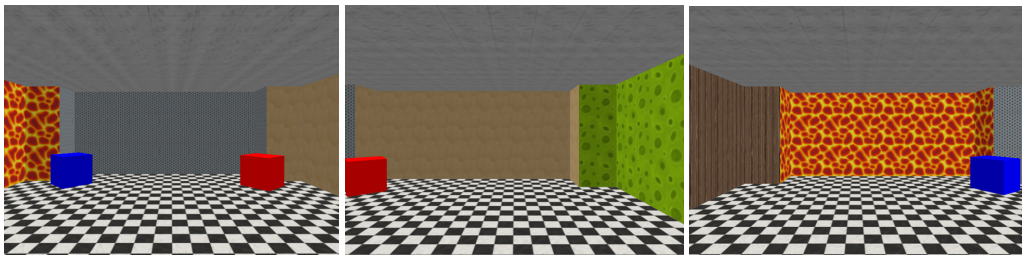


Figure 15: Maze 3D. The goal is either located at the blue or the red box. When the wall on the opposite side of the boxes (i.e. not observed in the leftmost image) has a wooden texture, the correct goal is the blue box, whereas if the texture is green, the red box is the goal.

The **Maze 3D** environment (Figure 15) is a continuous maze problem implemented using gym-miniworld (Chevalier-Boisvert, 2018), with 3 discrete actions (forward, left, right) where the objective is to reach one of the two possible goals, resulting in a reward of $+1$ (resp. -1) when the correct (resp. wrong) goal is reached. If a box is touched, the episode ends. The maze's axis range from -40 to 40, the two turn actions (*left*, *right*) modify the angle by 45 degrees, and the *forward action* is a 5 length move. The agent starts in a random position with a random orientation. The information about which goal to reach at each episode is encoded by the use of two different textures on the wall located on the opposite side of the boxes. In this way, the agent cannot simultaneously observe both boxes and the "informative" wall.

This environment allows to evaluate the models in a setting where the observation is a high dimensional space (3x60x60 RGB image). The mapping between the RGB image and the task target in $\{-1, 1\}$ is challenging and the informed policy should provide better auxiliary task targets than TI thanks to the "easy" training of the informed policy.

IMPORT outperforms TI on this environment (Figure 16) in both final performance and sample efficiency.

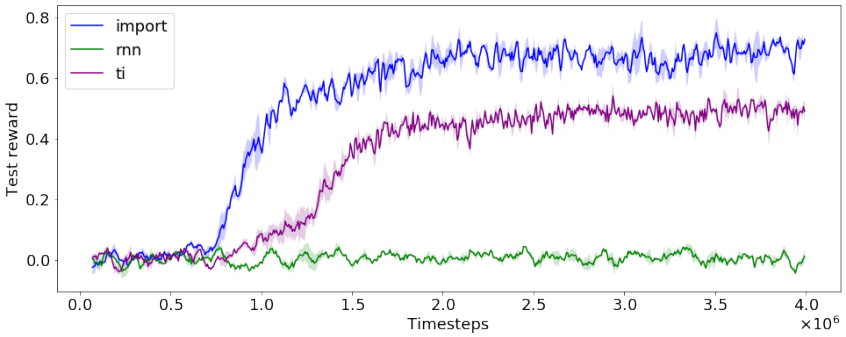
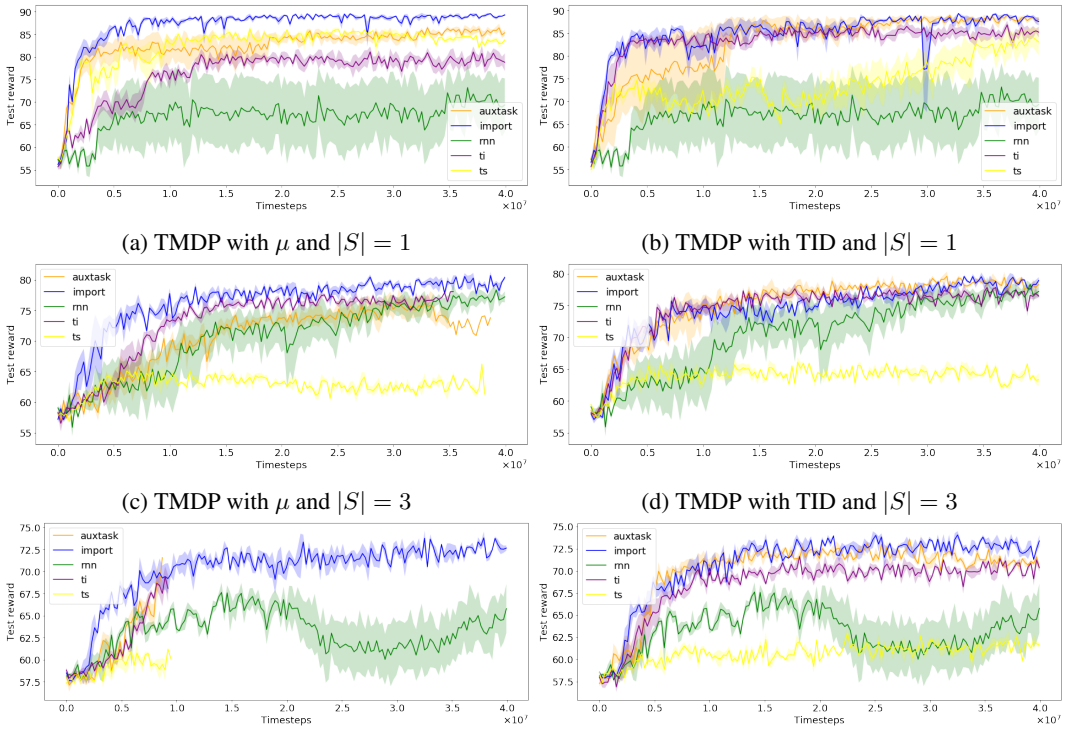


Figure 16: Learning curves on the Maze 3D environment

C.5 TABULAR MDPS

Tabular MDP (Duan et al., 2016) is a MDP with S discrete states and A actions such that the transition matrix is sampled from a flat Dirichlet distribution, and the reward function is sampled from a uniform distribution in $[0, 1]$. The task identifier μ is a concatenation of the transition and reward functions resulting in a vector of size $S^2A + SA$, allowing to test the models with high-dimensional μ.

IMPORT outperforms all baselines in all settings (Figure 17 and Table 2).



(a) TMDP with μ and $|S| = 1$

(b) TMDP with TID and $|S| = 1$

(c) TMDP with μ and $|S| = 3$

(d) TMDP with TID and $|S| = 3$

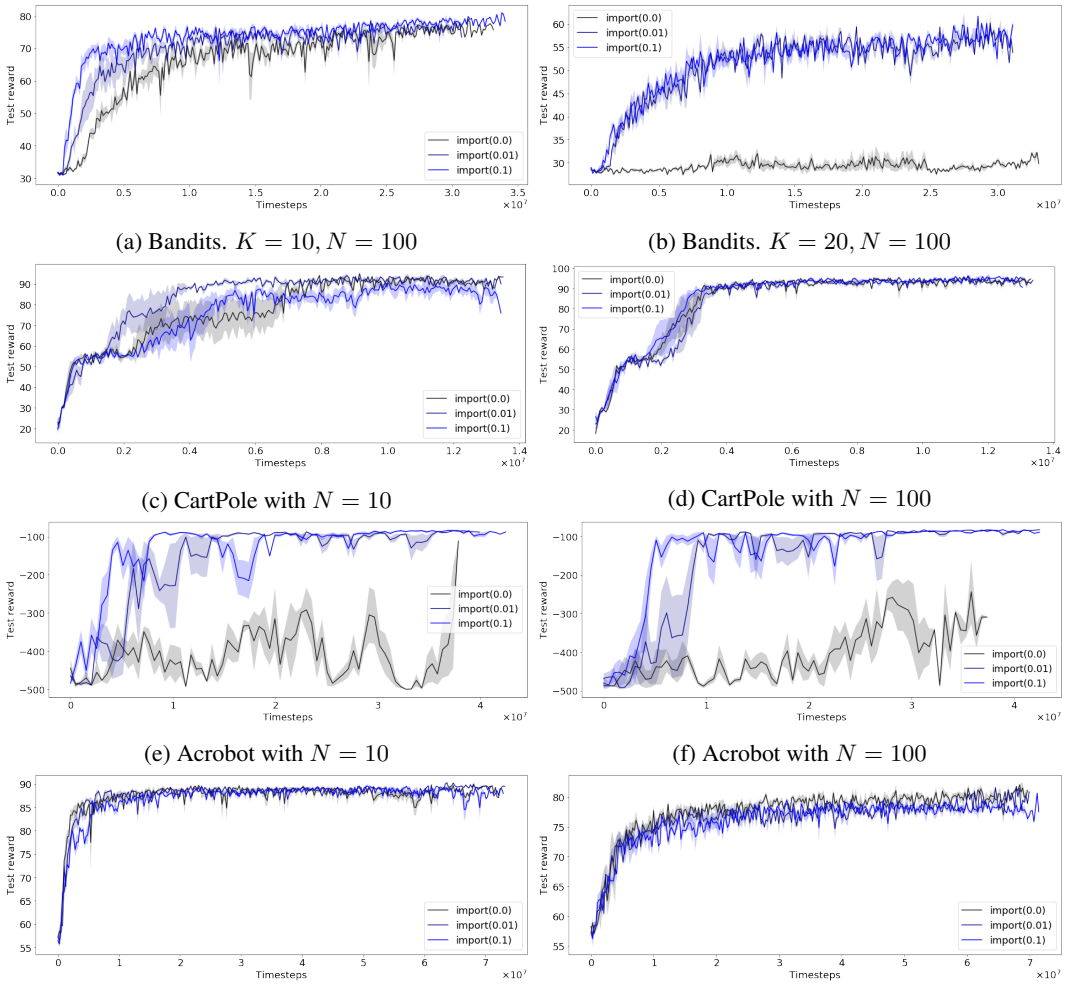
(e) TMDP with μ and $|S| = 5$

(f) TMDP with TID and $|S| = 5$

Figure 17: Evaluation on Tabular-MDP with different parameters and task descriptors (TID stands for task identifier).

D IMPACT OF THE β HYPERPARAMETER

We study the sensibility of the β parameter on IMPORT. Figure 18 clearly shows the benefits of using the auxiliary objective. On all but the Tabular-MDP environments, the recurrent policy successfully leverages the auxiliary objective to improve both sample efficiency and final performance for Acrobot.



(a) Bandits. $K = 10, N = 100$

(b) Bandits. $K = 20, N = 100$

(c) CartPole with $N = 10$

(d) CartPole with $N = 100$

(e) Acrobot with $N = 10$

(f) Acrobot with $N = 100$

(g) TMDP with $|S| = 1, |A| = 5, N = 100$

(h) TMDP with $|S| = 3, |A| = 5, N = 100$

Figure 18: Test performance of IMPORT for different β parameters (auxiliary supervised objective). We only report performance on informative μ task descriptors.