

Warm Starting Neural Dynamics Models From Few Jacobian Samples

Maxime Rochkoullets* Joppe De Jonghe** Mariya Ishteva**

* *Independent Researcher*

** *NUMA, KU Leuven, Belgium*

Abstract: Model-based reinforcement learning algorithms learn a dynamics model of their environment, often parameterized as a neural network and trained via backpropagation. When initialized randomly, the first stretch of training is spent learning coarse structure that the environment could have revealed directly. Motivated by evidence that good world model initialization improves the efficiency of these algorithms, we propose a cheap initialization scheme for one-hidden-layer multilayer perceptrons, assuming that gradient information is available at initialization. From few samples, we recover weights and hidden nonlinearities using a decoupling algorithm, and use these to warm-start the neural dynamics models. We show that in the low-data regime, our algorithm consistently outperforms random initialization and value-based pretraining, and either matches or outperforms Sobolev pretraining.

Keywords: model-based reinforcement learning, tensor decoupling, neural network initialization

1. INTRODUCTION

1.1 Model-Based Reinforcement Learning

Model-based approaches in reinforcement learning are a class of algorithms that leverage a (given or learned) model of their environment (M. Moerland et al., 2023). This simulation is often denoted as *world model* or *dynamics model*. This work focuses on the case where the model is learned rather than given, and we assume a known reward function.

Access to this model, if learned with sufficient accuracy, enables for *planning*: the ability to select future actions without interacting with the real world, by running a simulation of the future and selecting the action leading to the best outcome (Chua et al., 2018). Access to a dynamics model also enables the cheap generation of synthetic samples, and MBRL algorithms are generally regarded as more sample efficient than their model-free counterparts (Janner et al., 2019).

Under a Markov decision process (Puterman, 2014) with state space $\mathcal{S}$, action space $\mathcal{A}$, and deterministic transition function $\mathbf{t}(s_t, \mathbf{a}_t) = s_{t+1}$, the learned dynamics model is a function $\mathbf{f}_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ parametrized by θ and trained to approximate

$$\mathbf{f}_\theta(s_t, \mathbf{a}_t) \approx \mathbf{t}(s_t, \mathbf{a}_t), \quad (1)$$

that is, predicting some future state of the real environment given an input (state, action) pair. In this work we assume that $\mathcal{S} = \mathbb{R}^{d_S}$ and $\mathcal{A} = \mathbb{R}^{d_A}$, that is, states and actions are represented as real-valued vectors. For convenience, we use $\mathbf{x}_t \in \mathbb{R}^{d_S+d_A}$ to denote the concatenation of s_t and $\mathbf{a}_t$

$$\mathbf{x}_t = \text{concat}(s_t, \mathbf{a}_t) \text{ and } \mathbf{f}_\theta(\mathbf{x}_t) \equiv \mathbf{f}_\theta(s_t, \mathbf{a}_t). \quad (2)$$

In deep RL, dynamics models are parametrized as neural networks and trained via backpropagation. In general, and compared to other fields of deep learning that use *deep* networks, the neural network architectures used in reinforcement learning¹ are often shallow (Bjorck et al., 2021), and one-

hidden-layer multilayer perceptrons (MLPs) are widely used for solving deep reinforcement learning problems.

Finally, because the world-model is learned from the agent's *own* interactions with its environment, errors and biases can be exploited by the planner, which in turn determines which data is collected next. Providing the dynamics model with a better starting point has been shown to substantially improve sample efficiency (Clavera et al., 2019; Seo et al., 2022). We argue that the quality of the starting point therefore matters precisely in the low-data regime, where MBRL is expected to outperform.

1.2 Tensor Decoupling

Tensor decoupling (Dreesen et al., 2015) is a class of iterative algorithms for finding *decoupled representations* (Figure 1) of multivariate functions. It is based on the canonical polyadic (CP) decomposition (Kolda and Bader, 2009) of a tensor of stacked Jacobian samples. More precisely, given a target *black-box* vector-valued function $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n$, tensor decoupling algorithms seek an equivalent representation in the form

$$\mathbf{f}(\mathbf{x}) = \mathbf{W}_1 \mathbf{g}(\mathbf{W}_0^T \mathbf{x}), \quad (3)$$

with linear transformations $\mathbf{W}_0 \in \mathbb{R}^{m \times r}$, $\mathbf{W}_1 \in \mathbb{R}^{n \times r}$ and element-wise nonlinearities

$$\mathbf{g}(\mathbf{u}) = [g_1(u_1) \dots g_r(u_r)]^T, \quad (4)$$

where r denotes the *rank* of the decoupling, which also corresponds to the rank of the underlying CP decomposition.

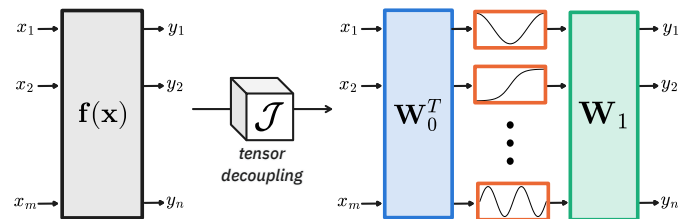


Fig. 1. Tensor decoupling algorithms map black-box functions into decoupled representations with flexible nonlinearities.

¹ This includes dynamics models in model-based RL, but also deep Q-networks and actor-critic networks in model-free algorithms.

This approach first requires collecting N Jacobian samples and stacking them into a 3-dimensional tensor

$$\mathcal{J} = \text{stack}(\mathbf{J}^{(1)} \dots \mathbf{J}^{(N)}) \in \mathbb{R}^{n \times m \times N},$$

$$\text{with } \mathbf{J} = \frac{\partial}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}) \in \mathbb{R}^{n \times m}. \quad (5)$$

Then, under certain uniqueness assumptions (Sidiropoulos et al., 2017), computing the CP decomposition of $\mathcal{J}$ yields the factors

$$\mathcal{J} = [\mathbf{W}_1, \mathbf{W}_0, \mathbf{G}], \quad (6)$$

where the columns of the third factor matrix $\mathbf{G} \in \mathbb{R}^{N \times r}$ contain the derivatives of our internals, evaluated in the sampled operating points. The final step consists in recovering the univariates in $\mathbf{g}$ by integrating in those points after curve fitting. Since all of the nonlinear behaviour of the target is captured by a small set of one-dimensional curves, this representation is often more interpretable than the initial black-box target.

Finally, one can make the observation that the decoupled representation (Figure 1) is a one-hidden-layer MLP with r hidden neurons and biases set to $\mathbf{0}$. The difference with classical MLPs being that the activations are flexible (see Apicella et al. (2021) for a taxonomy and equivalences), meaning that typically $g_i(x) \neq g_j(x)$ for $i \neq j$, compared to the more widely used MLPs with fixed activations (LeCun et al., 2015).

1.3 Contributions

Motivated by evidence that good world model initialization improves the efficiency of MBRL algorithms, we propose tdec-init (tensor decoupling initialization), an initialization scheme for dynamics models in MBRL algorithms. Assuming that transition Jacobians can be sampled at initialization, we show that this approach is both computationally cheap and particularly well suited for the low-data regime.

To the best of our knowledge, this work is the first to propose the use of tensor decoupling for initializing neural networks, and the first to compare tensor decoupling to other pretraining approaches in a reinforcement learning context.

The source code for our algorithm and experiments is available at <https://github.com/mrochk/tdec-init>.

2. RELATED WORK

2.1 Model Initialization in MBRL

In most MBRL implementations, the dynamics model is learned from scratch, initialized randomly with generic schemes designed for neural networks (LeCun et al., 2002; He et al., 2015). Several lines of work attempt to provide the model with an informed starting point.

The first family of approaches transfers knowledge from data other than the target environment. Clavera et al. (2019) meta-learn a dynamics model that can be rapidly adapted to new tasks from recent transitions, while Seo et al. (2022) pretrain world models before fine-tuning them with MBRL. These approaches substantially improve sample efficiency, but rely on other related tasks or external datasets.

A second family exploits prior knowledge of the target system itself. In nonlinear system identification, neural state-space models are commonly initialized from the best linear approximation of the system (Marconato et al., 2013), which improves convergence but only captures the linear part of the dynamics.

Finally, when derivatives are available, Sobolev training (Czarnecki et al., 2017; Zhang et al., 2025) fits a network to both the values and the Jacobians of the target function. It relies on gradient-based optimization of a non-convex objective, which in practice requires manually tuning the number of iterations, or monitoring a held-out set. Similar to Sobolev training, tdec-init leverages first-order information from the target environment. Unlike it, the weights and nonlinearities of the network are recovered directly from the tensor decomposition, without gradient-based pretraining.

2.2 Tensor Decoupling For Neural Networks

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3. METHODOLOGY

3.1 Setup

We assume a setting where first-order information about the dynamics is available before learning starts. After initialization, the agent only observes transitions, as in standard MBRL. In our experiments, Jacobians are computed by automatic differentiation of the true environment, from uniformly sampled state-action pairs.

3.2 Motivation

We believe that using tensor decoupling is well motivated for our task. This is due to the success of decoupling approaches in the field on nonlinear systems identification, which consists precisely on capturing the mathematical model from measurements of the system inputs and outputs. These systems are generally relatively compact in terms of number of inputs, outputs, and rank. This fits well the field of model-based reinforcement learning too, where small neural networks are often sufficient to model the dynamics of the environment (Bjorck et al., 2021). The mapping from decoupled representation to MLP is also direct, compared to deeper networks that would require more a sophisticated approach.

3.3 Decoupling Algorithm Implementation

This work uses a more recent tensor decoupling algorithm, called CMTF-BSD (De Jonghe and Ishteva, 2025). It is based on the same skeleton as described in Section 1.2 but additionally leverages zeroth-order information (Zniyed et al., 2021), which helps in guiding the algorithm towards sound representations even in non-unique CPD scenarios. It also uses B-splines to model its nonlinear internals instead of polynomials.

This work uses the implementation provided by the decoupling JAX library (Bradbury et al., 2018; Rochkoulets, 2026).

3.4 Algorithm

Algorithm 1 Tensor decoupling initialization (tdec-init)

Require: simulator with differentiable transition function

$\mathbf{t}(\mathbf{x}_t) = \mathbf{s}_{t+1}$, rank r , number of samples N

1: collect samples $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{J}^{(i)})$ from

$$\mathbf{y} = \Delta \mathbf{t}(\mathbf{x}) = \mathbf{t}(\mathbf{x}) - \mathbf{s}, \quad (7)$$

for $i = 1 \dots N$ and where $\mathbf{J} = \partial \mathbf{y} / \partial \mathbf{x}$

2: stack them into input matrix $\mathbf{X} \in \mathbb{R}^{N \times m}$, output matrix $\mathbf{Y} \in \mathbb{R}^{N \times n}$ and Jacobian tensor $\mathcal{J} \in \mathbb{R}^{n \times m \times N}$

3: $(\hat{\mathcal{J}}, \hat{\mathbf{Y}}) \leftarrow \text{scale}(\mathcal{J}, \mathbf{Y})$

4: compute the rank- r decoupling

$$(\mathbf{W}_0, \hat{\mathbf{W}}_1, \mathbf{g}) \leftarrow \text{decoupling}(\mathbf{X}, \hat{\mathbf{Y}}, \hat{\mathcal{J}}, r) \quad (8)$$

5: $\mathbf{W}_1 \leftarrow \text{unscale}(\hat{\mathbf{W}}_1)$

6: initialize bias vectors $\mathbf{b}_0 = \mathbf{b}_1 = \mathbf{0}$

7: initialize the dynamics model MLP

$$\mathbf{f}_\theta(\mathbf{x}) = \mathbf{W}_1 \mathbf{g}(\mathbf{W}_0^\top \mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1 \quad (9)$$

8: freeze the nonlinear internals $g_1 \dots g_r$

9: return $\mathbf{f}_\theta$ such that $\mathbf{f}_\theta(\mathbf{x}_t) + \mathbf{s}_t \approx \mathbf{s}_{t+1}$

The *scale* and *unscale* functions in Algorithm 1 refer to the tensor rescaling introduced in Rochkoulets et al. (2026). It is particularly important in this context, because the state space $\mathcal{S}$ is often made of components that have vastly different scales (e.g. position, velocity, angle), leading the decoupling algorithm to focus on the larger magnitude components.

Finally, by *freezing* the internals (line 6) we mean not updating them during subsequent training, rather than stopping the gradient-flow, as it would otherwise prevent us from updating the first linear layer $\mathbf{W}_0$.

3.5 Properties

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

4. EXPERIMENTS

We now compare different initialization schemes to tdec-init. First, we measure the accuracy of the world model *before* running the MBRL algorithm (identification accuracy), after after an additional pretraining of 10k steps. Second, we compare the returns of the agent in the environment when learning using a given initialization scheme (closed-loop performance).

4.1 Environments

We use 5 classic continuous control environments re-implemented in JAX when needed, so that transition Jacobians can be obtained via automatic differentiation. The environments used are Reacher from (Lenton et al., 2021), Pendulum, CartPole and Acrobot from Brockman et al. (2016), and the Planar Quadrotor given by (Tedrake, 2014).

Table 1 gives the dimensions of the state and action spaces, the maximum episode length T , the time step in second, the rank r of the decoupling, which also corresponds to the hidden width for all dynamics models, and the planning horizon H for our MBRL algorithm.

Table 1. Environments hyperparameters.

	d_S	d_A	T	step (s)	r	H	K
Reacher	4	2	100	0.05	4	20	1000
Pendulum	2	1	200	0.05	8	20	1000
CartPole	4	1	200	0.05	16	20	1000
Acrobot	4	1	500	0.2	16	20	1000
Quadrotor	6	2	200	0.02	16	40	1000

4.2 Dynamics Model

The dynamics model is a one-hidden layer MLP with biases. The activations are the splines returned by the tensor decoupling when tdec-init is used, and GELU activations (Hendrycks and Gimpel, 2016) with the baselines used for comparison. The second part of our first experiment (additional pretraining on identification accuracy) therefore also aims to measure the impact of using flexible (albeit frozen) activations, compared to the more classic static nonlinearities.

4.3 MBRL Algorithm

For the MBRL algorithm itself, we use model predictive control (MPC) with random shooting (Nagabandi et al., 2018). In our setting with deterministic environment, given reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and $\gamma = 1$, MPC corresponds to solving at each step t the following optimization problem

$$\mathbf{a}_{t:t+H-1}^* = \arg \max_{\mathbf{a}_{t:t+H-1}} \sum_{i=0}^{H-1} R(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}), \quad (10)$$

and because we use the learned world model for planning, it is approximated by

$$\hat{\mathbf{a}}_{t:t+H-1} = \arg \max_{\mathbf{a}_{t:t+H-1}} \left(R(\mathbf{s}_t, \mathbf{a}_t) + \sum_{i=1}^{H-1} R(\hat{\mathbf{s}}_{t+i}, \mathbf{a}_{t+i}) \right), \quad (11)$$

where $\hat{\mathbf{s}}_{t+i+1} = \hat{\mathbf{s}}_{t+i} + \mathbf{f}_\theta(\hat{\mathbf{s}}_{t+i}, \mathbf{a}_{t+i})$ corresponds to the prediction of the dynamics model. Random shooting approximately solves this optimization problem by sampling K action sequences uniformly from the action space and keeping the best

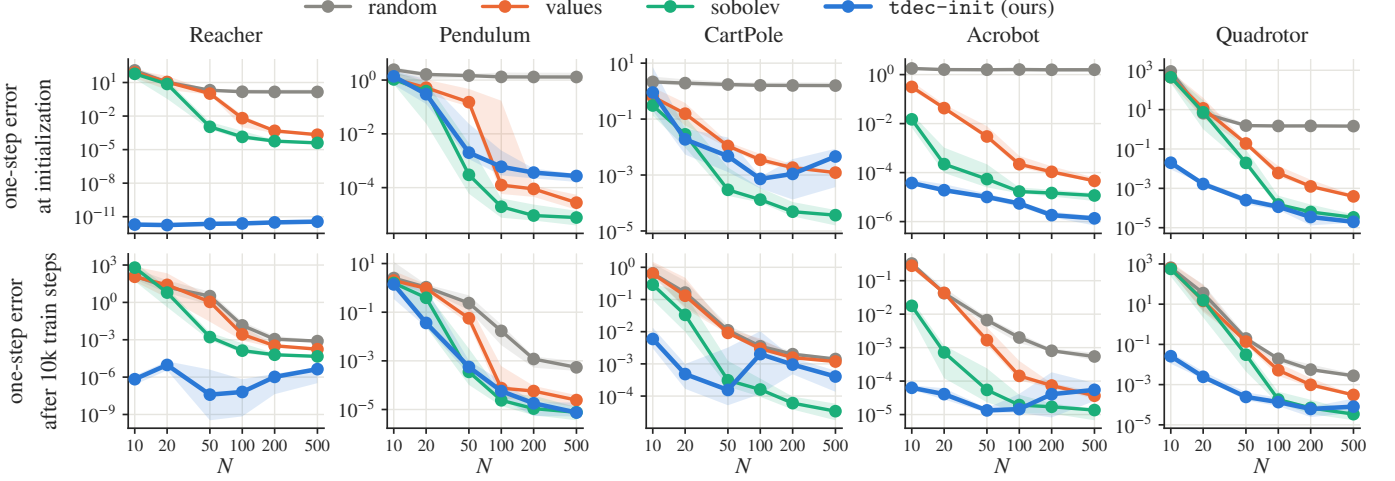


Fig. 2. One-step error for different initialization schemes as a function of the number of samples, measured before running the MBRL algorithm (identification accuracy), and after value pretraining for an additional 10k steps. In the low-data regime, our method consistently outperforms random initialization and value pretraining, and outperforms or matches Sobolev pretraining.

one. Only the action $\hat{a}_t$ is applied, and the same problem is solved again from the next state s_{t+1} . For all environments and baselines (including the oracle), K is set to 1000 (Table 1).

We pick this algorithm and planner because of their simplicity. It has no learned component other than the world model. Closed-loop performance therefore depends mostly on the accuracy of the dynamics model, isolating the effect of its initialization. Any other—potentially more sophisticated—planner could be used on top of the initialization schemes considered here.

4.4 Baselines

We compare our approach to 3 other initialization schemes, namely random initialization, value pretraining and Sobolev pretraining. Random initialization is LeCun normal initialization (LeCun et al., 2002) on standardized inputs with zero biases. Value pretraining corresponds to gradient-based pretraining (from random initialization) on input-output pairs with mean-squared error (MSE) on normalized targets. Finally, Sobolev pretraining (Czarnecki et al., 2017) is similar to value pretraining but adds the Jacobians in the objective with equal weight, making it the closest comparison since it uses the same information as tdec-init.

For value and Sobolev pretraining, we report the best performance out of fixed number of training steps (1k and 10k) and stopping defined by the validation loss not decreasing for 2k steps on a held-out validation set. This set consists of 20% of the given samples. The optimizer is Adam (Kingma and Ba, 2014) with learning rate 10^{-3} and batch size 32 (with replacement).

5. RESULTS

5.1 Identification Accuracy

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra

metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

5.2 Closed-Loop Performance

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

6. DISCUSSION

6.1 Remarks

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

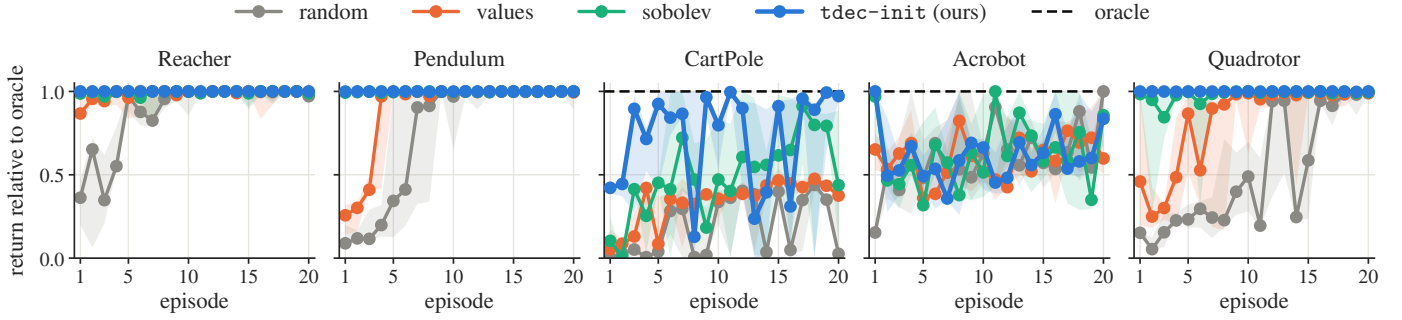


Fig. 3. Return of random-shooting MPC with different initialization schemes, relative to the oracle on the same episode, over the episodes of the MBRL algorithm (closed-loop performance). Our method matches the oracle from the first episode on Reacher, Pendulum and Quadrotor, and frequently reaches near-oracle returns on CartPole.

6.2 Future Work

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

ACKNOWLEDGEMENTS

Place acknowledgments here.

REFERENCES

- Apicella, A., Donnarumma, F., Isgrò, F., and Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, 138, 14–32.
- Bjorck, N., Gomes, C.P., and Weinberger, K.Q. (2021). Towards deeper deep reinforcement learning with spectral normalization. *Advances in neural information processing systems*, 34, 8242–8255.
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Katariya, Y., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs. URL <http://github.com/google/jax>.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.
- Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31.
- Clavera, I., Nagabandi, A., Liu, S., Fearing, R.S., Abbeel, P., Levine, S., and Finn, C. (2019). Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*. URL <https://openreview.net/forum?id=Hyzts0C5Y7>.
- Czarnecki, W.M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. (2017). Sobolev training for neural networks. *Advances in neural information processing systems*, 30.
- De Jonghe, J. and Ishteva, M. (2025). Non-parametric b-spline decoupling of multivariate functions. In *2025 33rd European Signal Processing Conference (EUSIPCO)*, 2117–2121. IEEE.
- Dreesen, P., Ishteva, M., and Schoukens, J. (2015). Decoupling multivariate polynomials using first-order information and tensor decompositions. *SIAM Journal on Matrix analysis and Applications*, 36(2), 864–879.
- He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.
- Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.
- Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32.
- Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kolda, T.G. and Bader, B.W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3), 455–500.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436–444.
- LeCun, Y., Bottou, L., Orr, G.B., and Müller, K.R. (2002). Efficient backprop. In *Neural networks: Tricks of the trade*, 9–50. Springer.
- Lenton, D., Pardo, F., Falck, F., James, S., and Clark, R. (2021). Ivy: Templated deep learning for inter-framework portability. *arXiv preprint arXiv:2102.02886*.
- M. Moerland, T., Broekens, J., Plaat, A., and M. Jonker, C. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1), 1–118.
- Marconato, A., Sjöberg, J., Suykens, J.A., and Schoukens, J. (2013). Improved initialization for nonlinear state-space modeling. *IEEE Transactions on instrumentation and Measurement*, 63(4), 972–980.
- Nagabandi, A., Kahn, G., Fearing, R.S., and Levine, S. (2018). Neural network dynamics for model-based deep reinforcement learning.

- ment learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, 7559–7566. IEEE.
- Puterman, M.L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.
- Rochkoulets, M. (2026). Decoupling: Fast tensor decoupling in JAX. URL <http://github.com/mrochk/decoupling>.
- Rochkoulets, M., De Jonghe, J., and Ishteva, M. (2026). Making tensor decoupling more robust and less parametric. *arXiv preprint arXiv:2610.16937*.
- Seo, Y., Lee, K., James, S.L., and Abbeel, P. (2022). Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, 19561–19579. PMLR.
- Sidiropoulos, N.D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E.E., and Faloutsos, C. (2017). Tensor decomposition for signal processing and machine learning. *IEEE Transactions on signal processing*, 65(13), 3551–3582.
- Tedrake, R. (2014). Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832). *Downloaded in Fall*.
- Zhang, X., Cai, X., Liu, B., Huang, W., Zhu, S.C., Qi, S., and Yang, Y. (2025). Differentiable information enhanced model-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 22605–22613.
- Zniyed, Y., Usevich, K., Miron, S., and Brie, D. (2021). A tensor-based approach for training flexible neural networks. In *2021 55th Asilomar Conference on Signals, Systems, and Computers*, 1673–1677. IEEE.