Appendix

I. Ablation Studies

A. Sampling Strategy

In order to study the effects of types of sampling strategies, we apply various sampling routines within MORLAX on a single environment, the Hopper.

1) `Dense`: Trade off vectors are sampled uniformly from the simplex using the Dirichlet distribution with $\alpha = 1$.
2) `Sparse`: Trade off vectors are sampled using MORLAX's chosen sampling method, described in Section III. We set $\kappa = m$ and vary clustering parameter K.

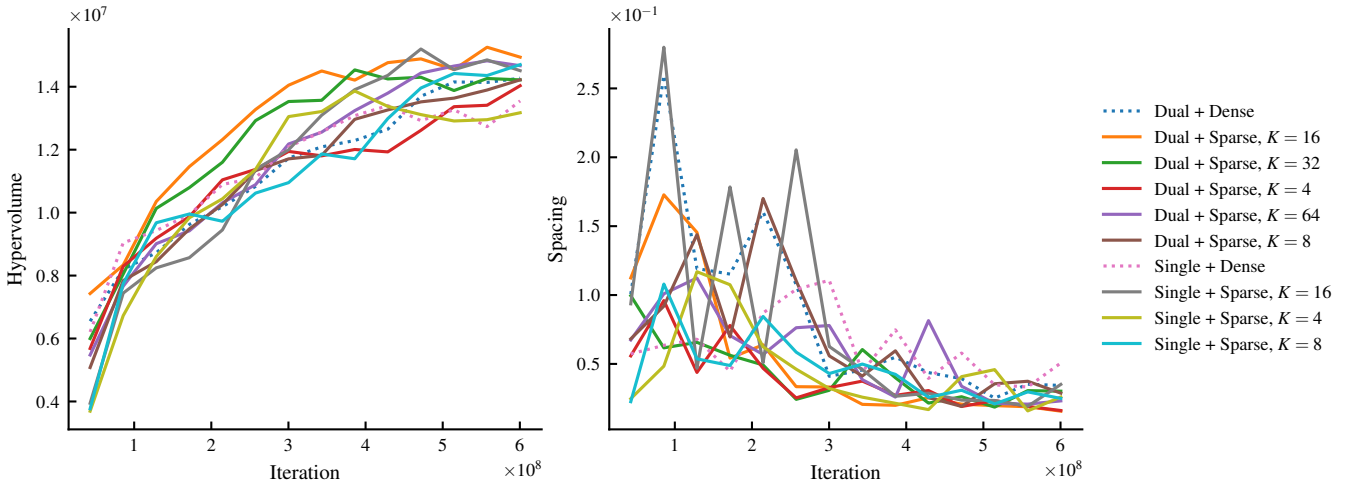

Fig. 1. Hypervolume by iteration comparison of the 2-objective MO-Playground Hopper environment across different sampling strategies and hypernetwork architectures. On average, the `Sparse` sampling strategy (for $K = 8, 16, 32, 64$) perform as well, or better than the `Dense` sampling strategy and the `Sparse` sampling strategy for $K = 4$ in terms of hypervolume. For all experiments, we set $\kappa = m = 2$.

As visualized in Fig. 1, the sparse sampling strategies with $K = 8, 16, 32, 64$ perform as well, or better than the dense sampling strategy, with $K = 16$ performing the best, in terms of hypervolume. We also find that the sparse sampling strategy with $K = 4$ generally underperforms the other strategies, indicating that the choice of K balances the diversity of the data with the stability of subsequent updates. Moreover, we observe this effect across different hypernetwork architectures, suggesting that the sparse sampling strategy is more effective at learning a diverse set of policies that can achieve high performance across a range of trade off vectors. To select K and κ, in future MORLAX experiments, we recommend setting $K \geq 8$ to encourage enough diversity in the data while maintaining relative stability, and setting κ to be equal to the number of objectives m. Notably, as the number of parallel environments increases, this may allow for larger values of K to be used. Note that in standard PPO, and in MORLAX, significantly increasing the number of parallel environments decreases the number of network updates per batch, potentially leading to slower learning. Thus, a balance must be struck between the number of parallel environments and the choice of K to ensure efficient learning while maintaining diversity in the data. We refer the reader to the implementation parameters in Table I for the number of parallel environments used in each MORLAX experiment, and generally recommend setting the number of parallel environments $N = 1024$.

B. Hypernetwork Architecture

To study the role of hypernetwork architecture, we apply two hypernetwork setups within MORLAX on a single environment, the Cheetah, which we modify to have 3 objectives instead of 2: speed, height, and energy consumption. This allows us to study the effects of hypernetwork architecture in a more complex setting than the 2-objective case. We compare the single hypernetwork architecture used by HYPER-MORL [16] with the dual hypernetwork architecture proposed by MORLAX.

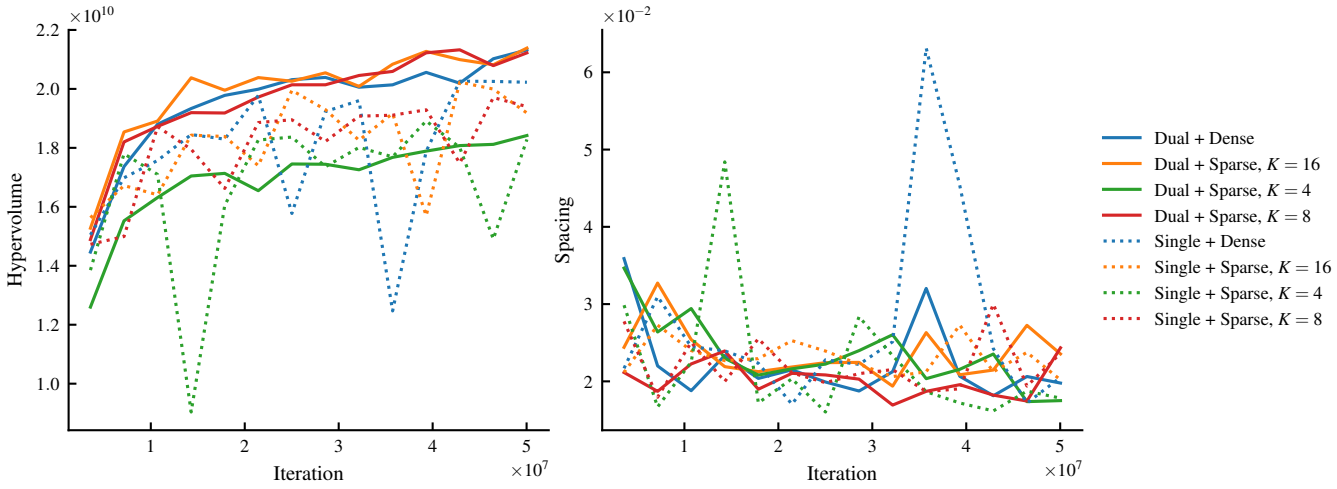

Fig. 2. Hypervolume by iteration comparison of the 3-objective MO-Playground Cheetah environmnent across different sampling strategies and hypernetwork architectures. On average, the dual hypernetwork architecture outperforms the single hypernetwork architecture, especially when paired with the sparse sampling strategy and $K \geq 8$. For all experiments, we set $\kappa = m = 3$.

We run all combinations of the two hypernetwork architectures and the two sampling strategies, with $K \in \{2, 4, 8, 16\}$ for the sparse sampling strategy and visualize these results in Fig. 2. We find that the dual hypernetwork architecture generally outperforms the single hypernetwork architecture, especially when paired with the sparse sampling strategy and $K \geq 8$. This suggests that the dual hypernetwork architecture is more effective at learning a diverse set of policies that can achieve high performance across a range of trade off vectors. Moreover, we observe that the dual hypernetwork architecture results in more stable learning than the single counterparts, as evidenced by the smoother learning curves. This may be because the dual hypernetwork architecture allows for more specialized representations of the policy and value functions, which can lead to more precise updates and learning.

Moreover, we observe in Fig. 1 that in the 2-objective case, the choice of hypernetwork architecture does not have as significant of an impact on performance, as the learning curves for both architectures are relatively similar per sampling strategy. We hypothesize that in the 2-objective case, the policy and value functions may not require as much specialization as in the 3-objective case, and thus the single hypernetwork architecture may be sufficient to learn a diverse set of policies that can achieve high performance across a range of trade off vectors.

Overall, these results suggest that the choice of hypernetwork architecture can have a significant impact on the performance of MORLAX, and that the dual hypernetwork architecture is preferred domains with many objectives.

II. Implementation Parameters

We include the implementation parameters for MORLAX in Table I and AMOR in Table II on the next page.

TABLE I

MORLAX IMPLEMENTATION PARAMETERS PER ENVIRONMENT.

Parameter	**MOCheetah**	**MOWalker**	**MOHopper**	**MOAnt**	**MOHumanoid**	**NaviGait**
PPO parameters						
action repeat	1	1	1	1	1	1
batch size	64	64	32	64	64	64
clipping epsilon	0.1	0.15	0.2	0.2	0.1	0.2
discounting	0.98	0.995	0.99	0.99	0.995	0.96
entropy cost	0.01	0.01	0.01	0.01	0.01	0.01
episode length	500	500	500	500	500	500
learning rate	5×10^{-4}	6×10^{-6}	5×10^{-6}	2×10^{-5}	2×10^{-6}	7×10^{-6}
max grad norm	1.0	1.0	1.0	1.0	1.0	1.0
normalize observations	true	true	true	true	true	true
num envs	128	1024	1024	1024	1024	2048
num evals	15	15	15	15	15	20
num minibatches	2	16	32	16	16	32
num timesteps	5×10^{7}	1.5×10^{8}	6×10^{8}	2×10^{8}	4×10^{8}	1.3×10^{8}
num updates per batch	4	4	4	4	4	4
reward scaling	1.0	1.0	1.0	1.0	1.0	1.0
unroll length	20	30	30	20	30	25
MORLAX parameters						
α	1.0	1.0	1.0	1.0	1.0	1.0
K	8	8	8	8	8	16
sampling	Sparse	Sparse	Sparse	Sparse	Sparse	Sparse
warmup fraction	0.0	0.0	0.0	0.0	0.0	0.0
hypernetwork type	dual	dual	dual	dual	dual	dual
hypernetwork hidden sizes	[128, 128]	[128, 128]	[128, 128]	[128, 128]	[512, 512]	[512, 512]
num features	16	16	16	16	32	64
policy hidden sizes	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[128, 128]	[256, 256]
value hidden sizes	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[128, 128]	[256, 256]

TABLE II

AMOR IMPLEMENTATION PARAMETERS PER ENVIRONMENT.

Parameter	**MOCheetah**	**MOWalker**	**MOHopper**	**MOAnt**	**MOHumanoid**
PPO parameters					
action repeat	1	1	1	1	1
batch size	64	64	32	64	64
clipping epsilon	0.1	0.15	0.2	0.2	0.1
discounting	0.98	0.995	0.99	0.99	0.995
entropy cost	0.01	0.01	0.01	0.01	0.01
episode length	500	500	500	500	500
learning rate	5×10^{-4}	6×10^{-6}	1×10^{-6}	2×10^{-5}	2×10^{-6}
max grad norm	1.0	1.0	1.0	1.0	1.0
normalize observations	true	true	true	true	true
num envs	128	1024	1024	1024	1024
num evals	15	15	15	15	15
num minibatches	2	16	32	16	16
num timesteps	5×10^{7}	4.5×10^{8}	6×10^{8}	4×10^{8}	8×10^{8}
num updates per batch	4	4	4	4	4
reward scaling	1.0	1.0	1.0	1.0	1.0
unroll length	20	30	30	20	30
AMOR parameters					
α	1.0	1.0	1.0	1.0	1.0
sampling	Dense	Dense	Dense	Dense	Dense
policy hidden sizes	[256, 256, 256]	[256, 256, 256]	[256, 256, 256]	[256, 256, 256]	[512, 512, 512]
value hidden sizes	[256, 256, 256]	[256, 256, 256]	[256, 256, 256]	[256, 256, 256]	[512, 512, 512]