

A FURTHER RESULTS

A.1 APPLYING DUTD TO PLANET

To demonstrate the generality of DUTD we additionally applied it to PlaNet Hafner et al. (2019) with the same hyperparameters for DUTD as we also used for DreamerV2. As base source code on which we implemented DUTD we used Pineda et al. (2021). We evaluated the resulting algorithm on three environments of the DeepMind Control Suite that were also used in the original publication of PlaNet. We used 5 seeds and evaluated the algorithms every 25000 environment frames. The results in Figure 9 show that DUTD also improves the performance of PlaNet. This is further evidence for the generality of DUTD.

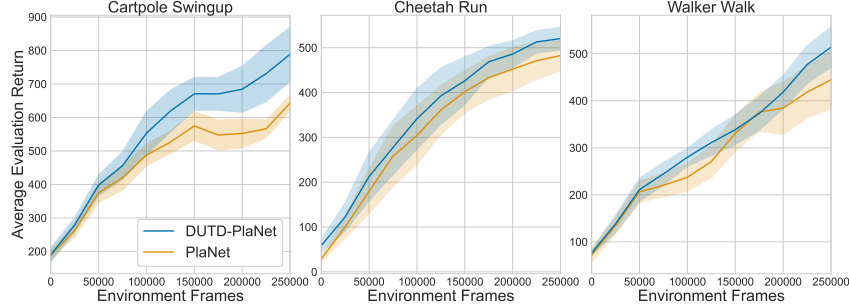


Figure 9: Learning curves for PlaNet with and without DUTD on three environments of the DeepMind Control Suite. The solid line is the mean over 5 seeds and the shaded area represents one pointwise standard deviation. We used a uniform filter of size 3.

A.2 DETAILED RESULTS FOR APPLYING DUTD TO DREAMERV2

The ten environments of the DeepMind Control Suite used to generate the aggregated curves in the Figures 3 and 6 are: acrobot.swingup, cheetah_run, finger_turn_easy, finger_turn_hard, hopper_hop, quadruped_run, quadruped_walk, reacher_hard, walker_walk, and walker_run.

We evaluated on all 20 environments used in the original Dreamer paper Hafner et al. (2020) but to save computation stopped training for ten environments at 1 million steps because standard Dreamer already reaches its asymptotic performance well before that mark. The aggregated curves are generated from the other 10 environments for which training ran until 2 million steps. Figure 12 shows the single learning curves for all environments. Please note, that on the 1 million steps environments with DUTD the asymptotic performance is reached much faster - often twice as fast.

In the Figures 10, 11, 12, 13, and 15 we present the more detailed results of our experiments for each single environment.

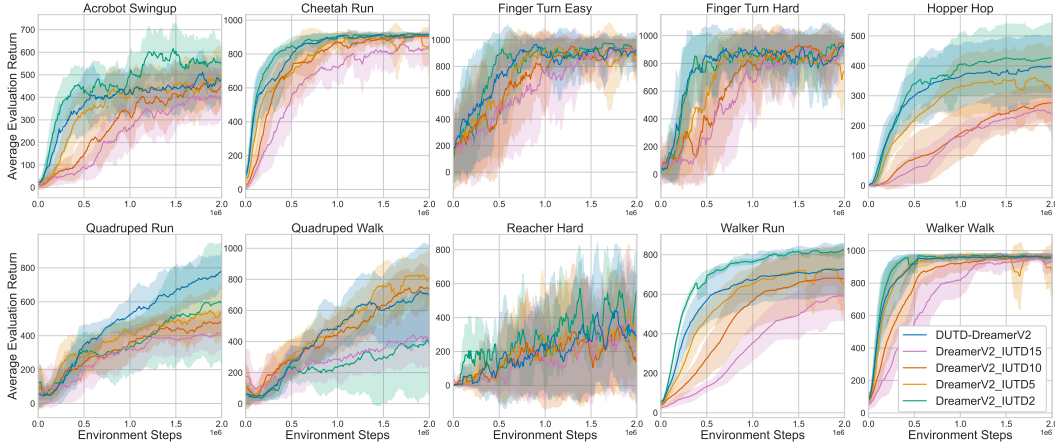


Figure 10: Learning curves for different choices of the IUTD ratio for each of the environments. The solid line is the mean over 5 seeds and the shaded area represents one pointwise standard deviation.



Figure 11: Learning curves for DreamerV2 with and without DUTD on the 26 environments of the Atari 100k benchmark. The solid line is the mean over 5 seeds and the shaded area represents one pointwise standard deviation.

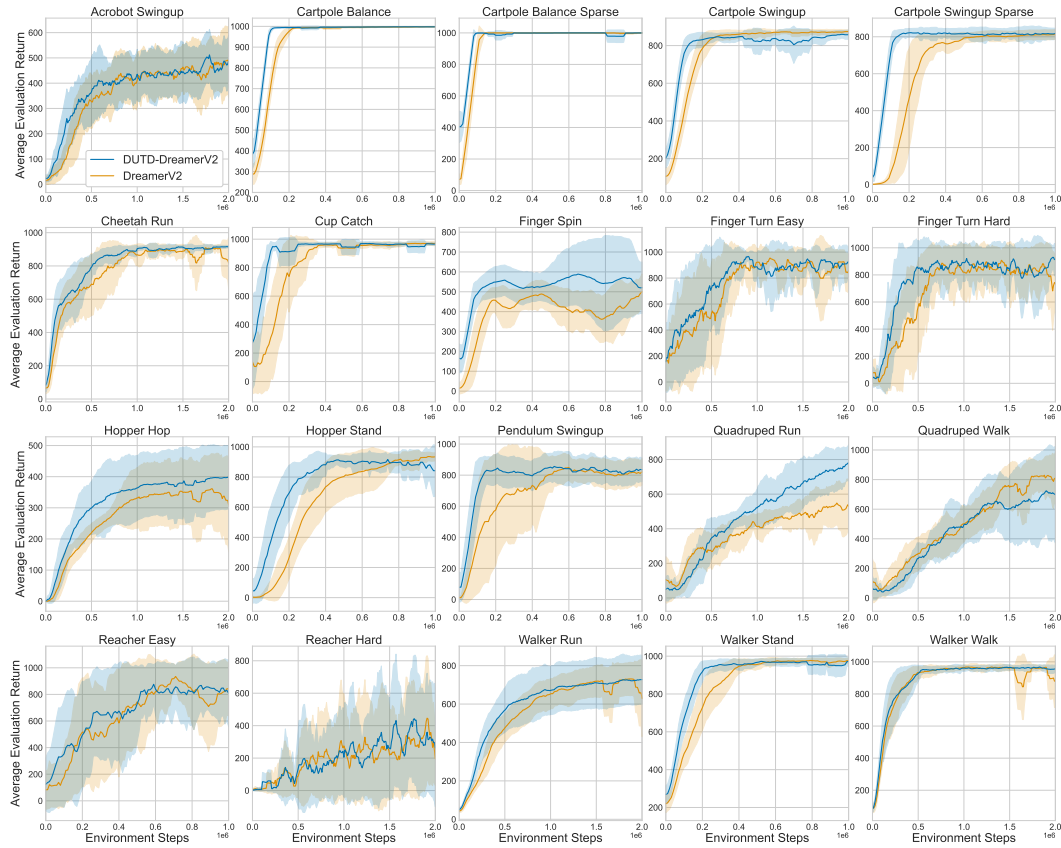


Figure 12: Learning curves for DreamerV2 with and without DUTD for 20 environments of the DeepMind Control Suite. The solid line is the mean over 5 seeds and the shaded area represents one pointwise standard deviation.

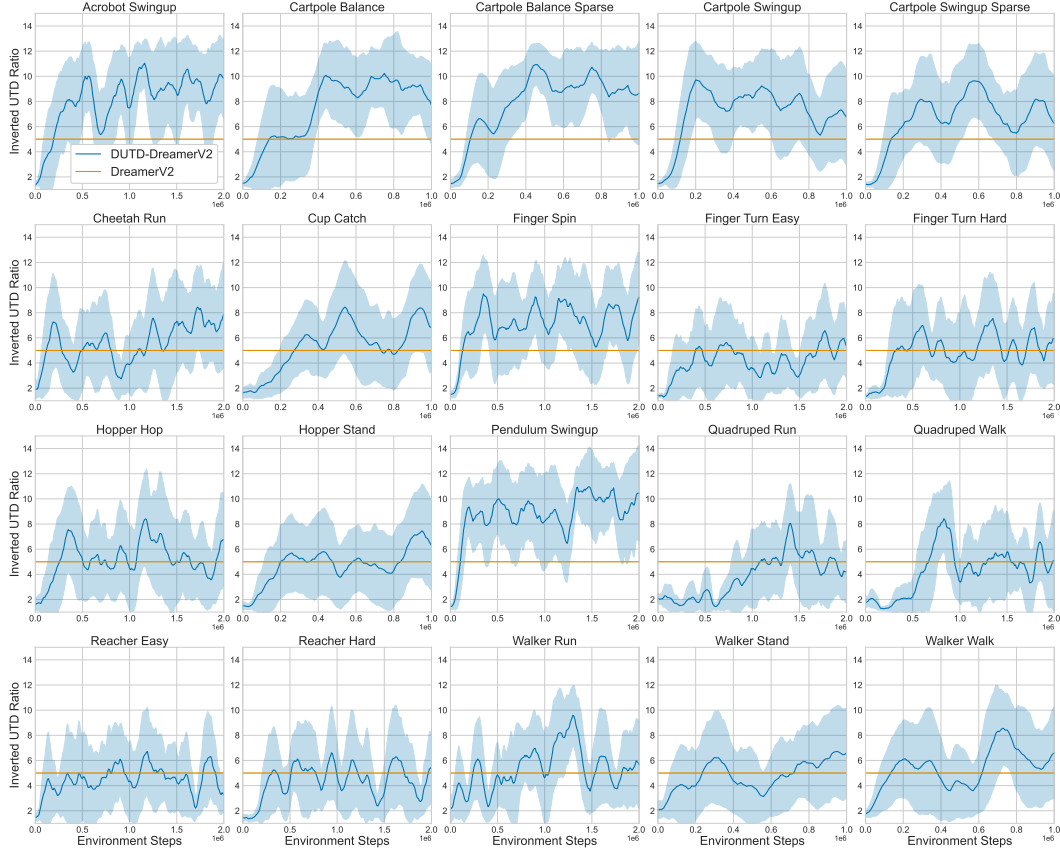


Figure 13: IUTD ratio against environment steps for DUTD and the standard DreamerV2 on all environments. For each environment the mean over 5 runs is plotted as the solid line and the shaded region shows represents one pointwise standard deviation in each direction.

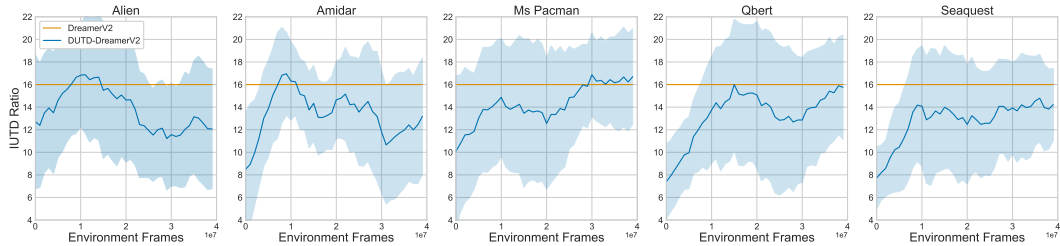


Figure 14: IUTD ratio against environment steps for DUTD and the standard DreamerV2 on 5 environments of Atari for which the algorithms were trained until 40 million frames. For each environment the mean over 3 runs is plotted as the solid line and the shaded region shows represents one pointwise standard deviation in each direction.



Figure 15: Learning curves for different choices of the IUTD ratio for each of the 26 environments of the Atari 100k benchmark. The solid line is the mean over 5 seeds and the shaded area represents one pointwise standard deviation.

B HYPERPARAMETERS

In Table 1 we give an overview of all hyperparameters related to DUTD. All other hyperparameters are the standard DreamerV2 hyperparameters as given in the open source codebase¹. On the DM Control Suite we reduced the number of steps d after which to collect new data for the validation set by a half during the first 400k steps as for some environments a strong policy is learned very quickly and hence a validation set with more recent transitions that better represent the kind of transitions the agent encounter makes more sense. We have because we started our first experiments with this but from some limited additional experiments it seems not to have a big impact on performance.

Table 1: Hyperparameters values for DUTD applied to DreamerV2 and the corresponding hyperparameter in the original DreamerV2.

HYPERPARAMETER	ATARI	DM CONTROL
INITIAL IUTD RATIO	16	5
LOWER BOUNDARY FOR THE IUTD RATIO	1	1
UPPER BOUNDARY FOR THE IUTD RATIO	32	15
IUTD UPDATE INCREMENT — c	1.3	1.3
NUMBER OF STEPS AFTER WHICH TO UPDATE THE IUTD RATIO — k	500	500
VALIDATION SET MAXIMUM SIZE — k	12,000	10,000
NUMBER OF STEPS AFTER WHICH TO COLLECT NEW DATA FOR THE VALIDATION SET — d	100,000	100,000
NUMBER OF ADDITIONAL TRANSITIONS FOR THE VALIDATION SET EACH TIME NEW VALIDATION DATA IS COLLECTED — s	3,000	3,000
	STANDARD DREAMERV2	
IUTD RATIO	16	5

C HYPERPARAMETER SENSITIVITY OF DUTD

Most hyperparameters of our method are straightforward to set and do not need any tuning. Updating the UTD ratio after the maximum episode length of 500 in DM Control Suite (DMC) is a value that we directly transferred to the Atari benchmark without further tuning. The initial value for the UTD ratio has no effect, as it gets quickly adjusted. The lower and upper limits for the UTD ratio are not reached often and hence do not affect performance given they are chosen lavish enough. We did not tune those. We tried a few choices for the number of additional transitions each time new validation data is collected and the number of steps after which we do so but did not find it to affect performance a lot and fixed one choice for both benchmarks.

The multiplicative factor c is the most important hyperparameter of our method and we hence conducted an additional experiment evaluating its sensitivity on the Atari100k benchmark over 5 random seeds. We show the aggregated metrics for different multiplicative factors in Figure 16.

The results show that seen over all metrics and relative to the baseline results the performance is not very sensitive with respect to the choice of the multiplicative factor. For the mean our default factor of 1.3 even gives slightly worse results than all other factors. Further, we argue the fact that the same setting of hyperparameters of DUTD works for very different benchmarks, Atari and DMC, shows that DUTD is not very sensitive to its hyperparameters and that the default values given by us will most likely work for a wide range of tasks. While an extensive hyperparameter search for the optimal UTD ratio might give slightly better results than DUTD with some fixed multiplicative factor, DUTD is still favorable for many real world applications where such tuning is too costly.

¹<https://github.com/danijar/dreamerv2>

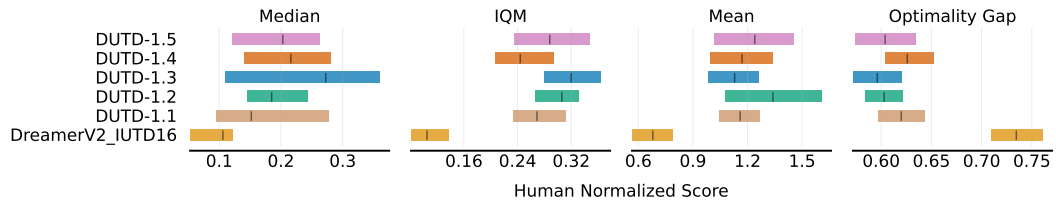


Figure 16: Aggregated metrics over 5 random seeds on the 26 games of Atari 100k, cf. Figure 2 for the methodology. We investigate the sensitivity of DUTD to its own most important hyperparameter c for values of 1.1, 1.2, 1.3 (default one used in the main experiments), 1.4, and 1.5 .