# A Pseudo-Code

**Algorithm 1** Open-Ended Neural Reward Functions

Initialize $\theta_0$ and $\psi_0$ and $O_{neg\_all}$.
$i \leftarrow 0$
**while** TRUE **do**
  $O_{neg_i}, O_{pos_i} \leftarrow \emptyset, \emptyset$
  Set $R_{\psi_i}$ as the reward of the MDP
  Train $\pi_{\theta_i}$ and $V_{\theta_i}$ using an actor-critic algorithm
  **for** $j = 0$ **to** $b$ **do**
    Reset the MDP to the initial state
    Follow $\pi_{\theta_i}$ for $k$ steps and add the observed states to $O_{neg}$
    Follow a random policy for $k'$ steps and add the observed states to $O_{pos}$
  **end for**
  $O_{neg\_all_i} \leftarrow O_{neg\_all_{i-1}} \cup O_{neg_i}$
  $\psi_{i+1} \leftarrow \psi_i$
  Train $\psi_{i+1}$ with $\mathcal{L}_\psi$, $O_{pos_i}$, $O_{neg_i}$ and $O_{neg\_all_i}$
  $\theta_{i+1} \leftarrow \theta_i$
  Set last layer of $\pi_{\theta_{i+1}}$ to 0
  $i \leftarrow i + 1$
**end while**

# B 2d Navigation: Experimental details

The episode length in the 2d navigation task is 250 steps. The guiding phase lasts for $2/3$ of the total steps of the 1-steps A2C agent learning. We sample the guiding length uniformly at random in the 0 to 200 range. For the negative samples we follow the learnt policy for 200 steps. For the positive samples, we take random actions for 50 steps after following the policy for 200 steps.

Table 3: Hyperparameters for the 2d navigation task.

| HYPERPARAMETER | |
|---|---|
| BASE ENTROPY REGULARIZATION | 0.005 |
| EXTRA ENTROPY REGULARIZATION | 0.05 |
| EPISODE LENGTH | 250 |
| A2C LEARNING RATE | 0.0001 |
| A2C DISCOUNT FACTOR | 0.99 |
| BATCH-SIZE | 2048 |
| STEPS PER SKILL | $2048 \cdot 60000$ |
| POSITIVE/NEGATIVE SAMPLE TARGET VALUE | $0.05/-0.05$ |
| REWARD NETWORK UPDATES PER SKILL | 500 |
| REWARD NETWORK LEARNING RATE | 0.001 |
| REWARD NETWORK TRAINING BATCH SIZE | $3 \cdot 256$ |

# C Robotic environments: Architecture and hyper-parameters

We use the PPO (Schulman et al., 2017) implementation from (Freeman et al., 2021) with modifications such that it allows our training method to work. In Table 4 and Table 5 we list the hyper-parameters we used. Note that we use a smaller learning rate for the bodies of the policy and value network. We found that this was beneficial for transferring more knowledge from the previous skill.

The environment specific parameters were found using Optuna (Akiba et al., 2019). For each environment we ran a search to optimize final performance on the environment rewards (running in positive $x$-direction). We used 200 runs and trained each of them for the same number of steps as a

Table 4: Hyperparameters for PPO shared in all three environments.

| BRAX PPO HYPERPARAMETERS | |
|---|---|
| POLICY HIDDEN LAYER SIZES | $[512, 512]$ |
| VALUE HIDDEN LAYER SIZES | $[512, 512]$ |
| BODY LEARNING RATE MULTIPLIER | 0.5 |
| EPISODE LENGTH | 1000 |
| ACTION REPEAT | 1 |
| NUMBER OF MINI-BATCHES | 32 |
| BATCH-SIZE | 1024 |
| PARALLEL ENVIRONMENTS | 2048 |

Table 5: Environment specific PPO hyperparemeters.

| | HALF-CHEETAH | ANT | HUMANOID |
|---|---|---|---|
| LEARNING RATE | 0.00010 | 0.00029 | 0.00017 |
| REWARD SCALING | 0.24532 | 5.58242 | 0.15326 |
| UNROLL LENGTH | 3 | 5 | 6 |
| UPDATES PER EPOCH | 15 | 14 | 10 |
| DISCOUNT FACTOR | 0.99109 | 0.92318 | 0.99114 |
| ENTROPY COST | 0.00062 | 0.00200 | 0.02087 |
| ENVIRONMENT STEPS PER SKILL | $10M$ | $10M$ | $20M$ |

skill in our method. This yielded hyper-parameters which are able to learn tasks in the corresponding environment. We did not take the downstream performance metrics (zero-shot performance and particle-based information) into account. Doing so would have exceeded our compute budget. In Table 6 the architecture and hyper-parameters used for the neural reward functions are listed. For the supervised reward training we use the Adam optimizer (Kingma & Ba, 2014). All code can be found in the supplementary material.

For DADS (Sharma et al., 2019) we used the code provided by the authors. For RND (Burda et al., 2018b) and 'Disagreement' (Pathak et al., 2019) we tuned the method specific hyper-parameters optimizing for the MI-metric of the $x$-velocity. In particular, for each environment and method we used $100$ runs of $200M$ environment steps per run.

## D Robotic environments: Zero-Shot Transfer

We use the Optuna hyperparameter tuning library (Akiba et al., 2019) to create Table 1. We ran multiple optimization procedures. For each one, we fix a number of environment steps, then optimize the final performance on the environment reward over $200$ runs. We iteratively increased the number of timesteps until the highest score of the $200$ runs beats our averaged zero-shot performance. We take the hyperparemters of that run and train $10$ agents until they outperform our averaged zero-shot performance. The average number of training steps needed for this is reported in Table 1. In the HUMANOID environment three of the runs did not reach the score in $100M$ steps, at which point we stopped training. We nonetheless took $100M$ into the average. The code for the hyperparameter search and the results of our conducted studies can be found in the supplementary material.

## E Robotic environments: Additional plots and Tables

In Figures 10, 11 respectively 12 we show more scatter plots for the ANT environment of our method, RND (Burda et al., 2018b) respectively 'Disagreement' (Pathak et al., 2019). In Figures 13, 14 respectively 15 we show more scatter plots for the HUMANOID environment of our method, RND respectively 'Disagreement'.

In Table 7 we report the zero-shot results for the ablations and the intrinsic curiosity baselines. In Table 8 we report the MI-metric results for the ablations.

Table 6: Neural reward function hyperparemeters for the BRAX environments.

| NEURAL REWARD FUNCTION HYPERPAREMETERS IN BRAX ENVIRONMENTS | |
|---|---|
| HIDDEN LAYER SIZES | [87] |
| HIDDEN LAYER NONLINEARITY | TANH |
| TARGET VALUE $a$ | 5 |
| GRADIENT STEPS | 300 |
| LEARNING RATE | 0.001 |
| TOTAL BATCH SIZE | 171 |
| NEGATIVE STEPS | 300 |
| POSITIVE STEPS | 40 |
| NUMBER OF SAMPLING ENVIRONMENTS | 8192 |
| FRACTION OF NEGATIVE SAMPLES STORED | 0.01 |

Table 7: Zero-shot environment reward of our algorithm, ablations and baselines.

| Task | Method reward | Zero-shot |
|---|---|---|
| Ant | **Ours (full)** | $\mathbf{2506 \pm 511}$ |
| Ant | Ours (policy ablation) | $1731 \pm 634$ |
| Ant | Ours (value ablation) | $2246 \pm 794$ |
| Ant | RND | $9 \pm 155$ |
| Ant | Disagreement | $-170 \pm 67$ |
| Humanoid | Ours (full) | $9092 \pm 1063$ |
| Humanoid | Ours (policy ablation) | $8906 \pm 616$ |
| Humanoid | Ours (value ablation) | $7357 \pm 816$ |
| Humanoid | RND | $7734 \pm 1916$ |
| Humanoid | **Disagreement** | $\mathbf{10107 \pm 736}$ |

Table 8: Ablations for the Particle-based mutual information metric. Results are averaged over 10 runs.

| Task | Method | MI$(s, z)$ |
|---|---|---|
| Ant | full method | $1.33 \pm 0.11$ |
| Ant | policy ablation | $1.09 \pm 0.16$ |
| Ant | value ablation | $1.28 \pm 0.15$ |
| Humanoid | full method | $1.29 \pm 0.25$ |
| Humanoid | policy ablation | $0.88 \pm 0.19$ |
| Humanoid | value ablation | $1.01 \pm 0.09$ |

# F Montezuma's Revenge: Adapting episode length

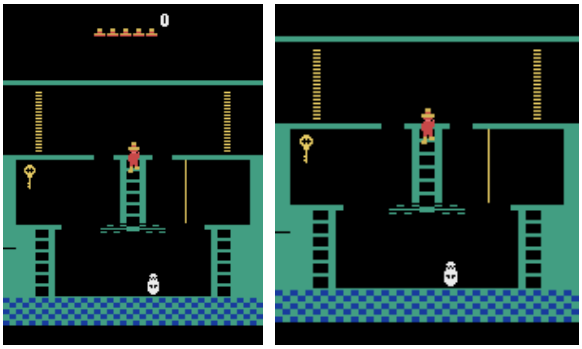
In order to save computation, we adapt the number of training steps according to several criteria. This lets us save a lot of compute on the skills which are easy to learn. We use the following measures:

- We train for $1M$ steps with guiding from the previous policy. The number of guiding steps is sampled uniformly at random between $0$ and $450$, each time the environment is reset. Then we train for another $0.65M$ steps without any guiding.
- If the average reward goes down after removing the guiding or is too low at any point, we restart the guiding phase for $0.65M$ steps.
- If the agent is reaching a terminal state in more than $10\%$ of the episodes, we continue training.

On top of this, we ignore positive samples that receive almost no reward. All these tricks enable us to considerably reduce the training time. However, they are not a core change in our algorithm as they could all be replaced by just training all generations for a longer fixed number of steps, just as before.

## G Montezuma's Revenge: Environment details

Figure 8 shows the initial state of Montezuma's Revenge and the cropped version of it that is used for our agent.



(a) (b)

Figure 8: (a) Initial state in the first room of Montezuma's Revenge. The agent controls the red/yellow character and its actions are moving up, down, right or left, and jumping. Touching the skull or jumping from a high height makes the agent lose a life. The agent must collect the key and open one of the two doors to access the next room. (b) Cropped version of the input that we use in our algorithm.

## H Montezuma's Revenge: Experimental details

In Montezuma's Revenge, we use the PPO implementation from the coax[11] library, with 1-step temporal differences. If the average score per $500$ steps is below $5$, we go back to the guiding phase. Due to the large memory requirements to store all negative samples, after the $15$-th epoch we rewrite old negative samples to add the new ones. Table 9 summarizes our hyper-parameters.

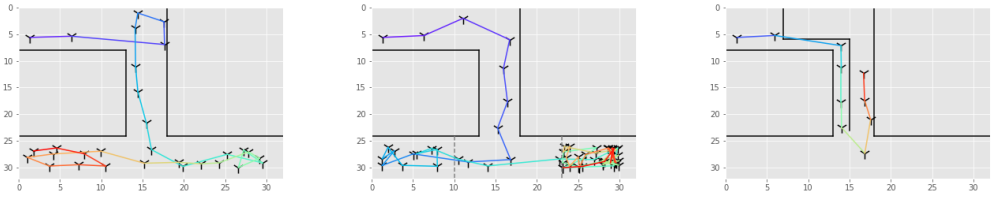
Table 9: Hyperparameters for Montezuma's Revenge.

| HYPER-PARAMETER | |
|---|---|
| BASE ENTROPY REGULARIZATION | 0.003 |
| EXTRA ENTROPY REGULARIZATION | 0.03 |
| EPISODE LENGTH | 500 |
| PPO LEARNING RATE | 0.0003 |
| PPO DISCOUNT FACTOR | 0.99 |
| PPO EPSILON | 0.2 |
| PPO BATCH SIZE | 1024 |
| PPO REPLAY BUFFER SIZE | 4096 |
| PPO REPLAY EPOCHS | 4 |
| PARALLEL ENVIRONMENTS | 32 |
| POSITIVE/NEGATIVE SAMPLE TARGET VALUE | $0.05/-0.05$ |
| REWARD NETWORK UPDATES PER SKILL | 1500 |
| REWARD NETWORK LEARNING RATE | 0.001 |
| REWARD NETWORK TRAINING BATCH SIZE | $3 \cdot 243$ |

[11] https://github.com/coax-dev/coax

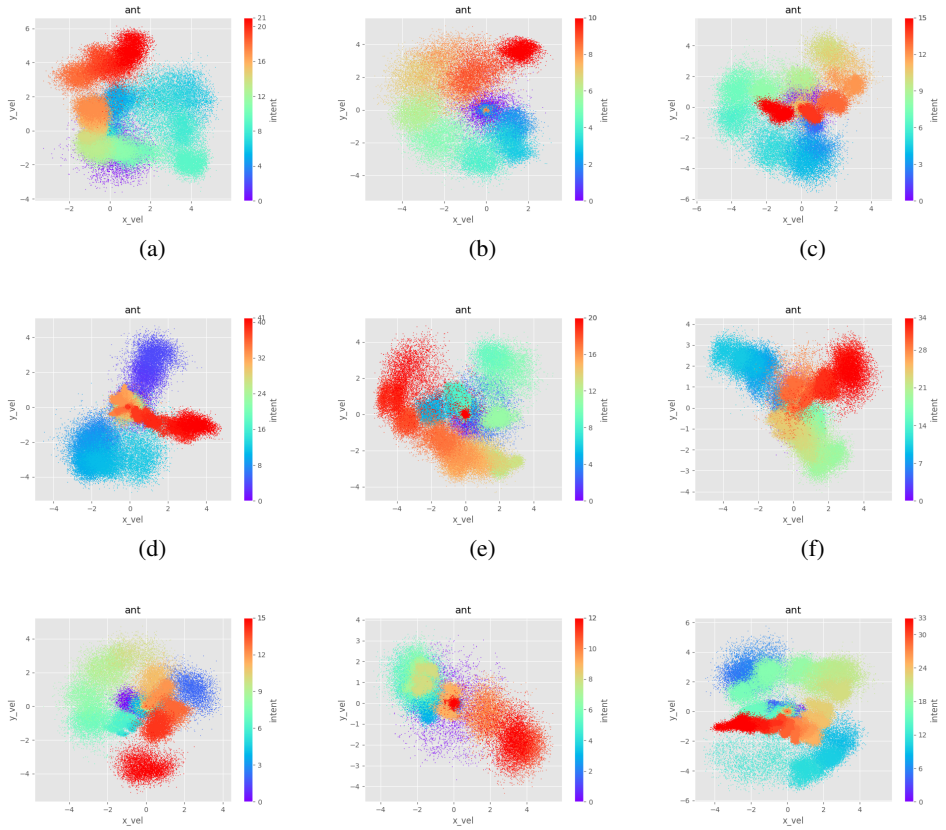
# I Additional 2d navigation experiments

In Figure 9 we show results of the 2d navigation task with three alternative environments. These environments illustrate the behavior of our algorithm in environments that require backtracking. In particular, Figure 9(b) shows a failure mode of our approach. A trap prevents the agent from backtracking and thus stops the progress of our algorithm once the search reaches that region.



(a) (b) (c)

Figure 9: The crosses correspond to the circles in Figure 2. (a) A T-Maze with two possible paths. The method backtracks when it reaches the end of the path and explores the other one. (b) The dotted line can not be crossed more than once in each episode. This traps the agent and makes it impossible to backtrack. The method gets stuck in one of the two traps. (c) A narrow path that leads to an intersection.




(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 10: Scatter-plots of the $x$- and $y$-velocity dimension of the states visited by the first $n$ skills of all additional 9 runs in ANT . $n$ was picked by hand in each run. The second half of the trajectory is shown. Colors change from early skills in purple to late skills in red.

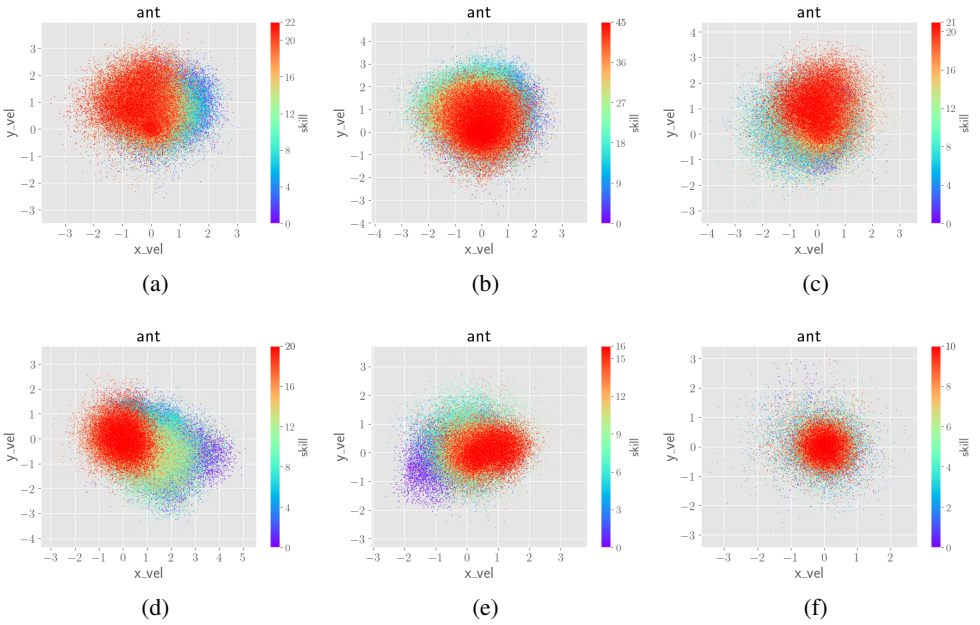



Figure 11: Scatter-plots of the $x$- and $y$-velocity dimension of the states visited by the first $n$ skills of 6 out of 10 RND runs in ANT . $n$ was picked by hand in each run. The second half of the trajectory is shown. Colors change from early skills in purple to late skills in red.

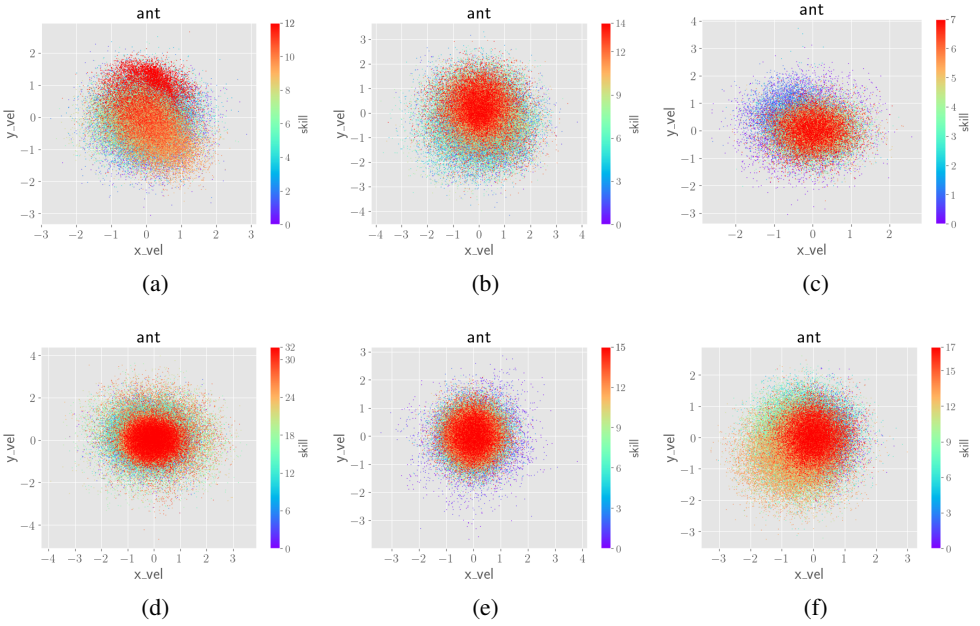



Figure 12: Scatter-plots of the $x$- and $y$-velocity dimension of the states visited by the first $n$ skills of 6 out of 10 'Disagreement' runs in ANT . $n$ was picked by hand in each run. The second half of the trajectory is shown. Colors change from early skills in purple to late skills in red.

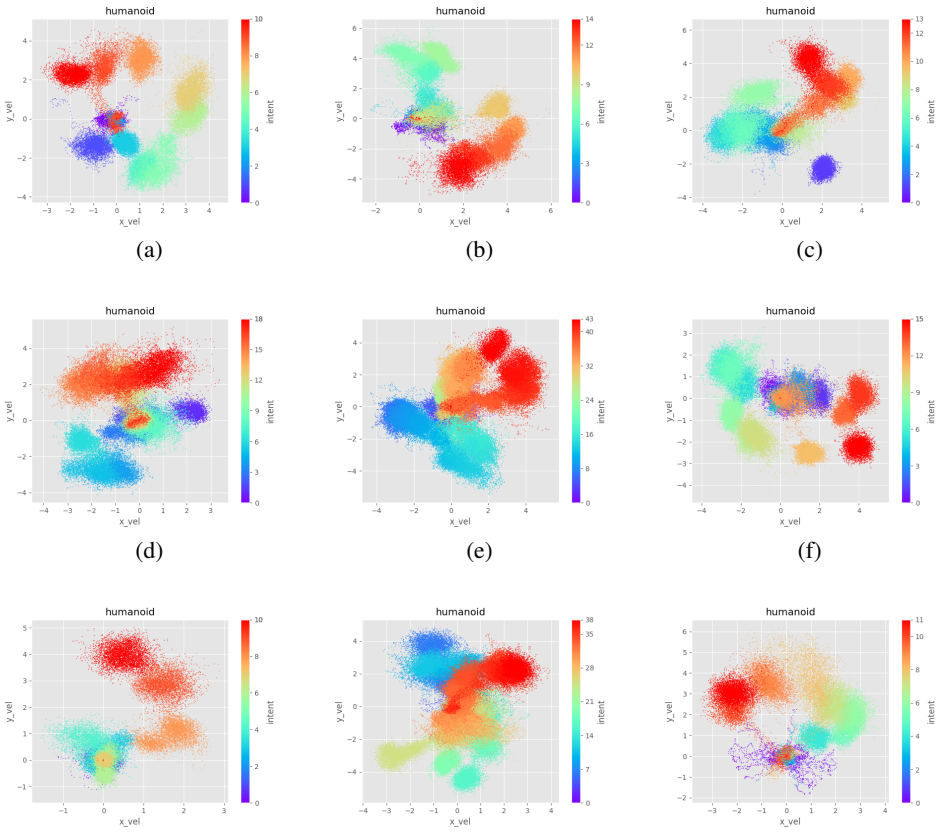



Figure 13: Scatter-plots of the $x$- and $y$-velocity dimension of the states visited by the first $n$ skills of all additional 9 runs in HUMANOID . $n$ was picked by hand in each run. The second half of the trajectory is shown. Colors change from early skills in purple to late skills in red.

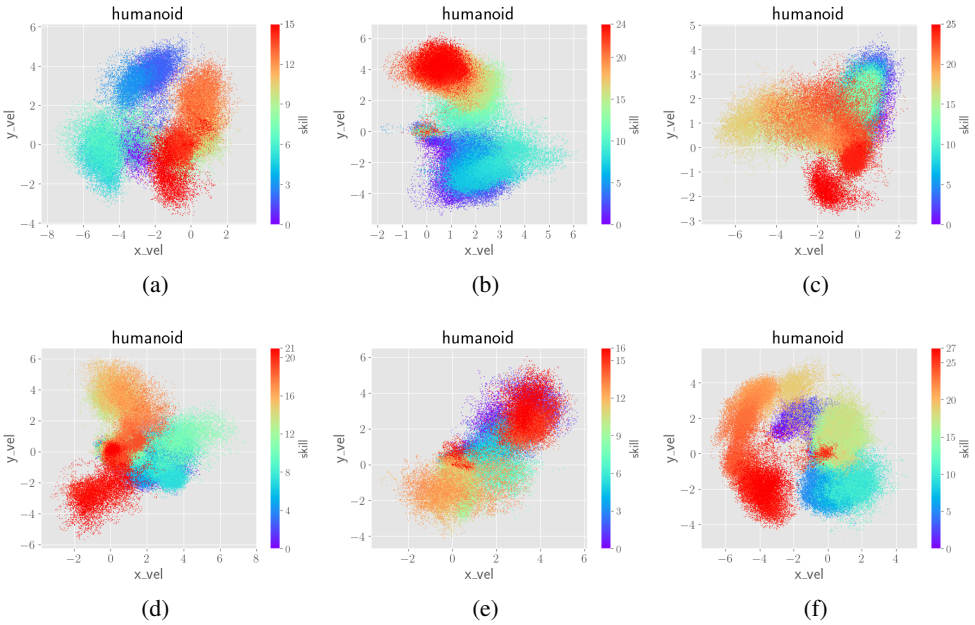



Figure 14: Scatter-plots of the $x$- and $y$-velocity dimension of the states visited by the first $n$ skills of 6 out of 10 RND runs in HUMANOID . $n$ was picked by hand in each run. The second half of the trajectory is shown. Colors change from early skills in purple to late skills in red.

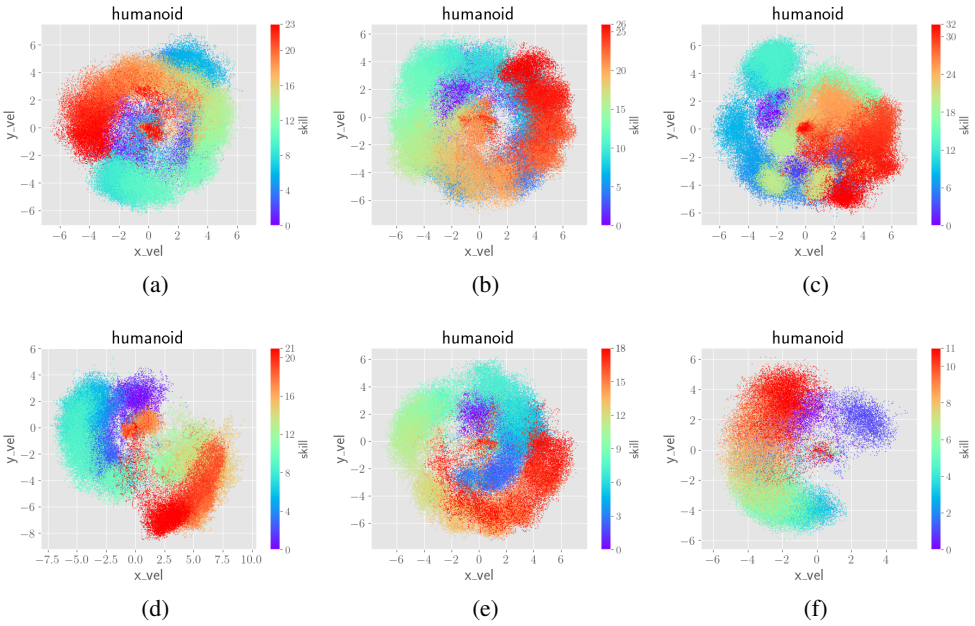



Figure 15: Scatter-plots of the $x$- and $y$-velocity dimension of the states visited by the first $n$ skills of 6 out of 10 'Disagreement' runs in HUMANOID . $n$ was picked by hand in each run. The second half of the trajectory is shown. Colors change from early skills in purple to late skills in red.