

Contrastive Decision Transformers – Supplementary Information

Sachin Konan*
Computer Science
Two Sigma Investments, LP
sachin@twosigma.com

Esmail Seraj
Interactive Computing
Georgia Institute of Technology
eseraj3@gatech.edu

Matthew Gombolay
Interactive Computing
Georgia Institute of Technology
matthew.gombolay@cc.gatech.edu

1 Algorithm Details and Pseudocode

In Algorithm 1, we define ConDT’s forward pass methods. In DTPRODUCT, we predict the next action from a history of K return, state, and action tokens, using a learnable return-dependent transformation of the state and action embeddings. This transformation is a diagonal matrix whose diagonal entries are populated by z_g . In DTSA, state and action embeddings are compressed into state-action embeddings with a linear layer, which is used in the training loops in Algorithm 2.

Algorithm 1 ConDT Forward Pass (for continuous actions)

▷ g, s, a, t : returns-to-go, states, actions, or timesteps
▷ embed_g : non-linear return embedding layer
▷ $\text{embed}_s, \text{embed}_a, \text{embed}_{sa}, \text{embed}_t, \text{pred}_a$: linear embedding layers
▷ transformer: GPT-2 Causal Transformer

```
1:
2: procedure EMBEDDINGS( $g, s, a$ )                                ▷ returns state and action embeddings
3:    $z_g \leftarrow \text{embed}_g(g)$ 
4:    $z_s \leftarrow z_g * \text{embed}_s(a)$ 
5:    $z_a \leftarrow z_g * \text{embed}_a(a)$ 
6:   return  $z_s, z_a$ 
7: end procedure
8:
9: procedure DTPRODUCT( $g, s, a, t$ )                               ▷ returns next predicted action
10:   $z_t \leftarrow \text{embed}_t(t)$ 
11:   $z_s, z_a \leftarrow \text{EMBEDDINGS}(g, s, a)$ 
12:   $z_s \leftarrow z_s + z_t$ 
13:   $z_a \leftarrow z_a + z_t$ 
14:   $z_{\hat{a}} \leftarrow \text{transformer}([z_s, z_a])$ 
15:   $\hat{a} \leftarrow \text{pred}_a(z_{\hat{a}})$ 
16:  return  $\hat{a}$ 
17: end procedure
18:
19: procedure DTSA( $g, s, a$ )                                       ▷ returns state-action embeddings
20:   $z_s, z_a \leftarrow \text{EMBEDDINGS}(g, s, a)$ 
21:   $z_{sa} = \text{embed}_{sa}([z_s, z_a])$ 
22:  return  $z_{sa}$ 
23: end procedure
```

*Research was conducted while as an undergraduate at Georgia Tech.

In Algorithm 2, we programmatically define the loop definitions of pre-training and parallel training, as well as the function definition of $\mathcal{L}_{\text{SimRCRL}}$. Pre-training trains the input embedding layers with $\mathcal{L}_{\text{SimRCRL}}$, followed by training all components of ConDT with \mathcal{L}_{DT} . Parallel training trains ConDT with $\mathcal{L}_{\text{ConDT}}$, which is a combination of \mathcal{L}_{DT} and $\beta * \mathcal{L}_{\text{SimRCRL}}$. We note that we publicly provide our codebase (including ConDT implementation, Algorithms 1-2, and the baselines) at <https://github.com/CORE-Robotics-Lab/ConDT>.

Algorithm 2 ConDT Parallel/Pre-Training Pseudocode

```

1: procedure SIMRCRL( $z_{sa}, \tau$ )
2:    $\mathcal{L}_{\text{SimRCRL}} = \sum_{i=0}^{\mathcal{B}_C} -\log \left( \frac{\exp((z_{sa}[i][0] \cdot z_{sa}[i][1])/\tau)}{\sum_{j=0}^{\mathcal{B}_C} \mathbb{1}(i,j) [\exp((z_{sa}[i][0] \cdot z_{sa}[j][0])/\tau) + \exp((z_{sa}[i][0] \cdot z_{sa}[j][1])/\tau)]} \right)$ 
3:   return  $\mathcal{L}_{\text{SimRCRL}}$ 
4: end procedure
5:
  ▷ pre-training:
6: for  $i = 1$  to pretrain_epochs do
7:    $g_c, s_c, a_c \leftarrow$  load contrastive data           ▷  $|g_c|, |s_c|, |a_c|: (\mathcal{B}_C, 2, *)$ 
8:    $z_{sa} \leftarrow$  DTSA( $g, s, a$ )
9:    $\mathcal{L}_{\text{SimRCRL}} \leftarrow$  SIMRCRL( $z_{sa}, \tau$ )
10:  Back-Propagate:  $\mathcal{L}_{\text{SimRCRL}}$ 
11: end for
12: for  $i = 1$  to train_epochs do
13:   $g, s, a, t \leftarrow$  load DT data                       ▷  $|g|, |s|, |a|, |t|: (\mathcal{B}, K, *)$ 
14:   $\hat{a} \leftarrow$  DTPRODUCT( $g, s, a, t$ )
15:   $\mathcal{L}_{\text{DT}} \leftarrow \|\hat{a} - a\|^2$ 
16:  Back-Propagate:  $\mathcal{L}_{\text{DT}}$ 
17: end for
18:
  ▷ parallel training:
19: for  $i = 1$  to train_epochs do
20:   $g_c, s_c, a_c \leftarrow$  load contrastive data           ▷  $|g_c|, |s_c|, |a_c|: (\mathcal{B}_C, 2, *)$ 
21:   $z_{sa} \leftarrow$  DTSA( $g, s, a$ )
22:   $\mathcal{L}_{\text{SimRCRL}} \leftarrow$  SIMRCRL( $z_{sa}, \tau$ )
23:   $g, s, a, t \leftarrow$  load DT data                       ▷  $|g|, |s|, |a|, |t|: (\mathcal{B}, K, *)$ 
24:   $\hat{a} \leftarrow$  DTPRODUCT( $g, s, a, t$ )
25:   $\mathcal{L}_{\text{DT}} \leftarrow \|\hat{a} - a\|^2$ 
26:  Back-Propagate:  $\mathcal{L}_{\text{DT}} + \beta * \mathcal{L}_{\text{SimRCRL}}$ 
27: end for

```

2 Environment and Dataset Details

Open-AI Gym – The Open-AI Gym environments, namely the hopper, the half-cheetah, and the walker2d, include three continuous-domain *balance* tasks shown in Table 1, top row. We present the state/action space specifics for each of these domains in Table 1, bottom row. In our experiments, each domain runs for a maximum of 1000 timesteps.

For each of these domains, we test on three variations of the dataset: the *Medium* (M.), the *Medium-Replay* (M.R.), and the *Medium-Expert* (M.E.) datasets. Details for each scenario are presented in the following:

1. *Medium* (M): Includes one million samples from a policy trained to achieve 1/3 of the performance of the expert.
2. *Medium-Replay* (M.R.): Utilizes the full replay buffer from the policy used to generate the Medium dataset. Timeouts are only marked when 1000 timesteps have been achieved.
3. *Medium-Expert* (M.E.): A 50/50 combination of the *Medium* dataset with samples from expert gameplay, resulting in around a total of two million samples.

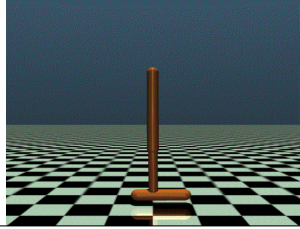
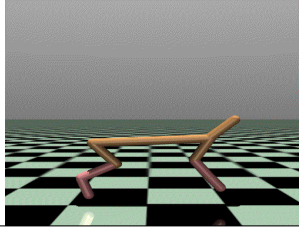
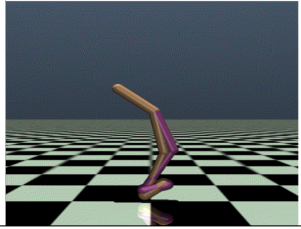
	hopper	half-cheetah	walker2d
			
$ \vec{S} $	11	17	17
$ A $	3	6	6

Table 1: Open-AI Gym environments utilized in our experiments.

Atari 2600 – These experiments include testing on four Atari environments. Each agent receives an RGB observation of size $(210 \times 160 \times 3)$, and takes a discrete actions that maximize return. Each Atari environment uses a frame-skipping parameter of four (i.e., repeating a selected action for four consecutive frames), and a stickiness parameter of 0% (i.e., the agent action is always taken). We present the action-space and maximum episode length specifics for these domains in Table 2.

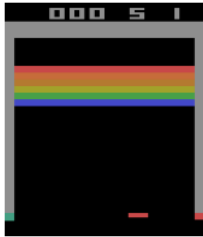
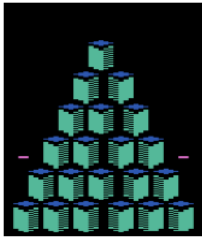
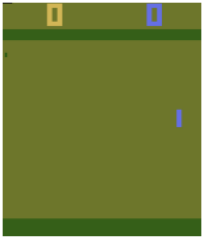

	Breakout	Qbert	Pong	Seaquest
				
$ A $	4	6	6	18
Max Steps	2654	3901	4731	2719
Goal	destroy all the bricks on the screen	change all the tile colors to a target color	hit pong ball as long as possible	avoid, collect, destroy objects at various depths

Table 2: Atari 2600 environments utilized in our experiments.

Adroit Hand-Gripping Domain – The Adroit Hand-Gripping Domain consists of a suite of environments wherein a simulated 24 degree-of-freedom (DoF) hand must perform hand-gripping tasks. The hands’ first, middle, and ring fingers have 4 DoF, pinky finger and thumb have 5 DoF, and wrist has 2 DoF. Each joint is controlled with an actuator and has an angle sensor to control position. There are three environments we sought to focus on:

1. Pen: A pen is randomly placed in an up-right configuration of the Adroit Hand, and the hand must shuffle the pen to match some target orientation.
2. Hammer: A hammer is placed on a table next to a board with a partially-stubbed nail. The Adroit Hand must pick up the hammer and repeatedly hammer the nail until it is pushed fully in.
3. Relocate: A ball is randomly placed on a table and the Adroit Hand must pick up the ball and place it in a random target area within the dimensions of the table and above its surface.

All these tasks have a reward function proportional to the percentage of the task completed; for instance, if the hand hammers the nail 50% in, half the maximum reward is given. Additionally, for each of these domains we decided to use a 75/25 split of trajectory data from a behavioral-cloning policy and expert policy. The behavioral-cloning policy data consists of 3750 trajectories generated by a policy trained on human demonstrations of the Adroit hand. The human demonstration were recorded using a human wearing a CyberGlove III, which allowed a human to control a physical Adroit Hand in each of the three scenarios. The expert policy data consists of 1250 trajectories generated by a policy trained on the Adroit simulator that achieved close-to-optimal reward performance. The mixture of these two policies allows the Decision Transformer to learn from both sub-optimal and optimal return performance, which reflects the potential composition of trajectory data with physical robotics. We present the state-space, action-space, and maximum episode length specifics for these domains in Table 3. We provide a video demo of the simulated Adroit Robotic HandGrip executing the learned policies by DT and ConDT for each task as supplementary material.

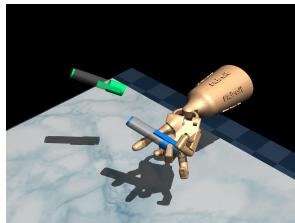

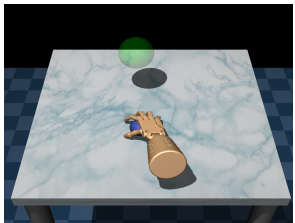
	Pen	Hammer	Relocate
			
$ \vec{S} $	45	46	39
$ A $	24	26	30
Max Steps	100	200	200

Table 3: Adroit Hand-Gripping Environments used in our experiments.

3 Hyper-Parameters

In Tables 3, 5, and 6, we present the training details for the Adroit, Atari, and Open-AI Gym experiments. For DT+Prod and ConDT, we utilize a learnable-dependent transformation, but since return, g_t , can be quite large at the beginning of the experiment, the transformation can be ineffective. For this reason, in some of the Gym, Atari, and Adroit Experiments, we scale g_t by the maximum target returns in each environment. Additionally, to train ConDT and ConDT w/o Prod, we either pre-train it’s embedding layers with $\mathcal{L}_{\text{SimRCRL}}$, followed by training the entire architecture with \mathcal{L}_{DT} , or parallel-train the entire architecture with $\mathcal{L}_{\text{ConDT}}$. For pre-training, we use a specific learning rate (Pre-Train lr) and train the entire architecture with (Main lr). Training the contrastive objective, $\mathcal{L}_{\text{SimRCRL}}$, is dependent on temperature (τ), batch size \mathcal{B}_C , and the size of the state-action embeddings ($|s_a|$). We found that a slightly larger state-action embedding size was required for Pong, because gradient magnitudes were too small when $|s_a| = 64$, like the other Atari experiments. For model hyper-parameters, we detail the number of transformer layers (N. Layers), the number of attention heads (N. Heads), the context length for the input sequence (K), and the transformers’ dropout rate (Dropout), in Table 4. We note that all model hyper-parameters remain unchanged, except for the context length used specifically in the Adroit experiments. Finally, for the Gym and Adroit experiments, we trained over 10 epochs with 10k and 5k iterations per epoch, respectively. For Atari, we trained over 10 epochs and the iterations is roughly the size of the training set in each environment divided by the batch size.

GPT Config	Gym	Atari	Adroit
N. Layers	3	6	3
N. Heads	1	8	1
K	20	30	5
Dropout	0.1	0.1	0.1

Table 4: GPT Hyper-parameters (Unchanged from original DT paper)

Env	Pre-Train	Scaled g_t	$ s_a $	Pre-Train lr	Main lr	\mathcal{B}	\mathcal{B}_C	β	τ
Breakout	✓	X	64	6e-4	6e-4	128	32	X	0.1
Qbert	X	✓	64	X	6e-4	128	32	0.1	0.1
Pong	✓	X	256	6e-4	6e-4	512	64	X	0.1
Seaquest	X	✓	64	X	6e-4	128	32	0.1	0.1

Table 5: Hyper-parameters for Atari Experiments

Dataset	Env	Pre-Train	Scaled g_t	$ s_a $	Pre-Train lr	Main lr	\mathcal{B}	\mathcal{B}_C	β	τ
M	hopper	✓	✓	50	6e-3	1e-4	64	64	X	0.1
M	halfcheetah	X	✓	50	X	1e-4	64	64	0.01	0.1
M	walker2d	✓	✓	50	6e-3	1e-4	64	64	X	0.1
M.R.	hopper	✓	✓	128	6e-3	1e-4	64	64	X	0.1
M.R.	halfcheetah	✓	✓	128	6e-3	1e-4	64	64	X	0.1
M.R.	walker2d	✓	✓	128	6e-3	1e-4	64	64	X	0.1
M.E.	hopper	✓	✓	128	6e-3	1e-4	64	64	X	0.1
M.E.	halfcheetah	✓	✓	128	6e-3	1e-4	64	64	X	0.1
M.E.	walker2d	✓	✓	128	6e-3	1e-4	64	64	X	0.1

Table 6: Hyper-parameters for Open-AI Gym Experiments

4 Execution Gain

In this section we investigate the change in training time with our optimizations on DT. In terms of change in model size, DT+Prod is actually smaller than DT because the input sequence becomes 2/3 of the original size because the return is pre-encoded to the transformer. ConDT w/o Prod is the same size as DT, but it has an extra linear projection layer to convert the state and action

embeddings to state-action embeddings. ConDT bears the model size changes of ConDT w/o Prod and DT + Prod, which together is a relatively negligible change in model size to DT. In terms of training time, ConDT does require an increase in training/evaluation time over DT because of the addition of an additional loss function. We detail the increase in training times of our methods as well as the average % gain of ConDT with and w/o pretraining in Table 7. Each entry in the table represents the combined execution time of training and evaluation, where evaluation is conducted after each epoch of training. Note that in some of the environments, finishing an evaluation faster is optimal, whereas in others, finishing an evaluation in a longer time is optimal. For example, in the Gym experiments, which generally have the largest execution time percentage gain, the agent is rewarded for staying upright for as long as possible, so naturally ConDT, which performs the best in all the Gym scenarios, has the longest execution time because agents stay upright longer than DT. Therefore, having a larger combined evaluation and training time does not necessarily imply suboptimal training in all of the experiments. Nevertheless, ConDT and ConDT with pretraining require 23% and 38% longer execution times across the different experiments. We believe this gain can be drastically reduced by parallelizing evaluation between epochs, and using a faster dataloader for the SimRCRL loss (which is the primary bottleneck with using our contrastive loss).

Experiment	DT	DT+Prod	ConDT w/o Prod	ConDT	ConDT Pretrain	ConDT % Gain	ConDT Pretrain % Gain
hopper-medium	145	100	130.15	164	231	13.10	59.31
halfcheetah-medium	244.18	167.8	201.53	242.6	304.55	-0.65	24.72
walker2d-medium	163.92	157.13	181.13	216.43	271	32.04	65.33
hopper-replay	154.82	214.47	220.53	237.9	277.95	53.67	79.54
halfcheetah-replay	268.33	288.5	348.28	345.69	374.3	28.83	39.49
walker2d-replay	187.03	212.1	243.48	240	269.92	28.32	44.32
hopper-expert	181.35	191.27	211.15	206.07	274.22	13.63	51.21
halfcheetah-expert	255.2	257.9	304.68	306.5	319.9	20.10	25.35
walker2d-expert	221.22	219.73	255.98	280.05	306.78	26.60	38.68
Pong	372.6	311.9	461.83	X	326	X	-12.51
Breakout	587.68	565.95	589.99	602.21	615.55	2.47	4.74
Qbert	348.7	306.95	482.78	X	326.7	X	-6.31
Seaquest	253.33	168.98	318.22	489.02	359.92	93.04	42.07
Pen	41.62	41.33	51.4	50.82	76.06	22.11	82.76
Hammer	92.61	66.02	75.42	73.13	97.68	-21.03	5.48
Relocate	62.45	64.1	70.52	68.62	67.8	9.88	8.57
Average	260.26	243.28	303.83	251.65	274.76	23.01	38

Table 7: Combined Training and Evaluation Time (in minutes) across Baselines and Experiments