A Model Architecture

In this section, we provide comprehensive details about the Transformer model architectures considered in this work. We implement all models in PyTorch [61] and adapt the implementation of Transformer-XL from VPT [4].

A.1 Observation Encoding

Experiments conducted on both DMLab and RoboMimic include RGB image observations. For models trained on DMLab, we use a ConvNet [29] similar to the one used in Espeholt et al. [20]. For models trained on RoboMimic, we follow Mandlekar et al. [53] to use a ResNet-18 network [29] followed by a spatial-softmax layer [23]. We use independent and separate encoders for images taken from the wrist camera and frontal camera. Detailed model parameters are listed in Table A.1.

Table A.1: Model hyperparameters for vision encoders.

Hyperparameter	Value
DMLab	
Image Size	72 × 96
Number of ConvNet Blocks	1
Channels per Block	[16, 32, 32]
Output Size	256
RoboMimic	
Image Size	84 × 84
Random Crop Height	76
Random Crop Width	76
Number of Randomly Cropped Patches	1
ConvNet Backbone	ResNet-18 [29]
Output Size	64
Spatial-Softmax Number of Keypoints	32
Spatial-Softmax Temperature	1.0
Output Size	64

Since DMLab is highly partially observable, we follow previous work [20, 22, 4] to supply the model with previous action input. We learn 16-dim embedding vectors for all discrete actions.

To encode proprioceptive measurement in RoboMimic, we follow Mandlekar et al. [53] to not apply any learned encoding. Instead, these types of observation are concatenated with image features and passed altogether to the following layers. Note that we do not provide previous action inputs in RoboMimic, since we find doing so would incur significant overfitting.

A.2 Transformer Backbone

We use Transformer-XL [16] as our model backbone, adapted from Baker et al. [4]. Transformer-XL splits long sequences into shorter sub-sequences that reduce the computational cost of attention while allowing the hidden states to be carried across the entire input by attending to previous keys and values. This feature is critical for the long sequence inputs necessary for cross-episodic attention. Detailed model parameters are listed in Table A.2.

A.3 Action Decoding

To decode joystick actions in DMLab tasks, we learn a 3-layer MLP whose output directly parameterizes a categorical distribution. This action head has a hidden dimension of 128 with ReLU activations. The “Goal Maze” and “Irreversible Path” tasks have an action dimension of 7, while “Watermaze” has 15 actions. To decode continuous actions in RoboMimic, we learn a 2-layer MLP that parameterizes a Gaussian Mixture Model (GMM) with 5 modes that generates a 7-dimensional action. This network

Table A.2: Model hyperparameters for Transformer-XL.

Hyperparameter	Value (DMLab)	Value (RoboMimic)
Hidden Size	256	400
Number of Layers	4	2
Number of Heads	8	8
Pointwise Ratio	4	4

has a hidden dimension of 400 with ReLU activations. During deployment, we employ the "low-noise evaluation" trick [31].

B Training Details and Hyperparameters

All experiments are conducted on cluster nodes with NVIDIA V100 GPUs. We utilize DDP (distributed data parallel) to accelerate the training if necessary. Training hyperparameters are listed in Table A.3.

Table A.3: Hyperparameters used during training.

Hyperparameter	Value (DMLab)	Value (RoboMimic)
Learning Rate	0.0005	0.0001
Warmup Steps	1000	0
LR Cosine Annealing Steps	100000	N/A
Weight Decay	0.0	0.0

C Experiment Details

C.1 DMLab Main Experiment

Our DMLab main experiment is conducted on three levels with task IDs

- `explore_goal_locations_large`,
- `rooms_watermaze`,
- and `skymaze_irreversible_path_hard`.

We use no action repeats during training and evaluation. For experiments with varying task difficulty, we select difficulty parameters "room numbers", "spawn radius", and "built-in difficulty" for these three levels, respectively. We adopt environment wrappers and helper functions from Petrenko et al. [63] to flexibly and precisely maneuver task difficulties.

Due to different task horizons, we tune the context length of Transformer-XL models and vary curricular trajectories accordingly. These differences are summarized in Table A.4.

RL oracles serve as source agents used to generate training data for our methods and the "BC w/ Expert Data" baseline. They are trained with the PPO [70] implementation from Petrenko et al. [63]. The "BC w/ Expert Data" baselines have the same model architecture, training hyperparameters, and amount of training data as our method, but are trained solely on trajectories generated by the best performing RL oracles without cross-episodic attention.

C.2 DMLab Generalization

This series of experiments probe the zero-shot generalization capabilities of embodied agents in unseen maze configurations, out-of-distribution difficulty levels, and varying environment dynamics. For the task "Goal Maze w/ Unseen Mechanism", we use the level with task ID

Table A.4: Experiment details on DMLab tasks. Columns "Epoch" denote the exact training epochs with best validation performance. We select these checkpoints for evaluation. For task-difficulty-based curriculum, the column "Training Trajectories" with n × m entries means n trajectories per difficulty level (m levels in total). The column "Sampled Episodes" with [i, j] entries means we first determine the number of episodes per difficulty level by uniformly sampling an integer from [i, j] (inclusively).

Level Name	Context Length	Task-Difficulty-Based Curriculum			Learning-Progress-Based Curriculum		
		Epoch	Training Trajectories	Sampled Episodes	Epoch	Training Trajectories	Sampled Episodes
Goal Maze	500	84	100 x 3	[1, 5]	88	300	9
Watermaze	400	89	100 x 3	[1, 5]	80	300	9
Irreversible Path	1600	90	100 x 4	[1, 3]	97	400	8

Table A.5: Evaluation results on DMLab, averaged over three tasks (Figure 3).

Ours (Task Difficulty), Auto	Ours (Task Difficulty), Fixed	Ours (Learning Progress)	DT (Mixed Difficulty)	DT (Single Difficulty)	AT (Mixed Difficulty)	AT (Single Difficulty)	BC w/ Expert Data	RL (Oracle)	Curriculum RL (Oracle)
51.4	54.4	32.4	35.3	11.7	42.7	33.4	14.2	40.6	50.6

Table A.6: Generalization results on DMLab, averaged over five settings (Figure 4).

Ours (Task Difficulty)	Ours (Learning Progress)	DT (Mixed Difficulty)	DT (Single Difficulty)	AT (Mixed Difficulty)	AT (Single Difficulty)	BC w/ Expert Data	RL (Oracle)	Curriculum RL (Oracle)
39.6	27.8	31.8	13.6	39.4	29.2	18.1	30.0	37.6

`explore_obstructed_goals_large`, which adds randomly opened and closed doors into the maze while ensuring a valid path to the goal always exists. An example of an agent's ego-centric observation is visualized in Figure A.1.

The task "Irreversible Path (OOD. Difficulty)" corresponds to configurations with the built-in difficulty of 1 (agents are only trained on difficulty up to 0.9, as noted in Table 1). For tasks with varying environment dynamics, we directly test agents with an action repeat of 2. This is different from the training setting with no action repeat.

C.3 RoboMimic Main Experiment

We leverage the Multi-Human (MH) dataset from Mandlekar et al. [53]. It consists of demonstrations collected by operators with varying proficiency. We construct the expertise-based curriculum by following the order of "worse operators, okay operators, then better operators". We use a context length of 200 for both tasks. There are 90 trajectories per expertise level. To determine the number of trajectories per expertise level when constructing curricular data, we uniformly sample an integer from $[1, 5]$ (inclusively). The "Lift" and "Can" tasks are solved after training for 33 epochs and 179 epochs, respectively. We control for the same number of training epochs in subsequent ablation studies.

C.4 Ablation Study on Curriculum Granularity

We perform this ablation with the task-difficulty-based curriculum on DMLab levels due to the ease of adjusting granularity. The definition of varying levels of curriculum coarseness is listed in Table A.7.

Table A.7: Definitions of varying levels of curriculum coarseness.

Level Name	Difficulty Parameter	Test Difficulty	Fine	Medium	Coarse
Goal Maze	Room Numbers	20	$5 \rightarrow 10 \rightarrow 15$	$5 \rightarrow 10$	$5 \rightarrow 15$
Watermaze	Spawn Radius	580	$150 \rightarrow 300 \rightarrow 450$	$150 \rightarrow 300$	$150 \rightarrow 450$
Irreversible Path	Built-In Difficulty	0.9	$.1 \rightarrow .3 \rightarrow .5 \rightarrow .7$	$.1 \rightarrow .5 \rightarrow .7$	$.1 \rightarrow .3 \rightarrow .5$

Figure A.1: A visualization of the task "Goal Maze (Unseen Mechanism)". It includes doors that are randomly opened or closed.

Table A.8: Results show the performance of different curricula on two robotic manipulation tasks: Lift and Can. Standard deviations are included.

Task	Expertise-Based Curriculum	Learning-Progress-Based Curriculum	CQL [41]
Lift	100.0 ± 0.0	32.0 ± 17.0	2.7 ± 0.9
Can	100.0 ± 0.0	30.0 ± 2.8	0.0 ± 0.0
Average	100.0	31.0	1.4

C.5 Comparison of Curricula in RoboMimic

In IL settings, we further explored the efficacy of various curricula. For the RoboMimic tasks examined, we employed a learning-progress-based curriculum, ensuring the total training trajectories matched those of the expertise-based curriculum (i.e., 270 trajectories per task). All other parameters remained consistent, with the training data derived from RoboMimic's machine-generated dataset.

Table A.8 indicates that when heterogeneous-quality human demonstrations are accessible, the expertise-based curriculum is preferable due to its superior performance over the learning-progress-based approach. Conversely, without expert demonstrations and relying solely on machine-generated data, the learning-progress-based curriculum is still commendable. It offers noteworthy results and surpasses offline RL methods like CQL [41], even though CQL is trained on the full RoboMimic dataset, encompassing 1500 trajectories for the Lift task and 3900 for the Can task.

D Feasibility of Obtaining Curricular Data

The challenge of accurately orchestrating a curriculum is non-trivial and hinges on various factors. In the present work, three curriculum designs are introduced and validated, each with its practical considerations and underlying assumptions, discussed herein.

Learning-Progress-Based Curriculum. RL agents typically exhibit monotonic improvement over training epochs, thereby naturally producing incrementally better data. The curriculum here is devised through a series of checkpoints throughout the training duration, necessitating no supplementary assumptions for its formulation.

Task-Difficulty-Based Curriculum. In contexts where environmental difficulty is parameterizable, curricula can be structured through a schedule, determined by the relevant difficulty parameter, as demonstrated within this work. In scenarios lacking parameterized difficulty, alternatives such as methods proposed by Kanitscheider et al. [40] may be employed. The application of our method to tasks where difficulty is not explicitly characterized presents an intriguing avenue for future research.

Expertise-Based Curriculum. A notable limitation resides in the requisite to estimate demonstrators' proficiency. While some IL benchmarks, e.g., RoboMimic [53], come pre-equipped with proficiency labels, a broader application of our method necessitates an approximation of proficiency. One plausible approach entails ranking trajectories via completion time. Furthermore,

a demonstrator's proficiency is likely to organically improve—from initial unfamiliarity with teleoperation systems or tasks, to a stage of executing data collection with muscle memory [52]. This progression potentially provides a rich learning signal conducive for CEC application.

E Broader Impact

Our Cross-Episodic Curriculum can significantly enhance Transformer agent learning but carries potential societal impacts. The efficiency of our method depends on the curriculum's design. If the curriculum unintentionally reflects biases, it could lead to the amplification of these biases in learned policies, potentially perpetuating unfair or discriminatory outcomes in AI-driven decisions. Furthermore, the computational intensity of our approach at evaluation could contribute to increased energy usage, which has implications for the environmental footprint of AI applications.